

The image is a large, symmetrical, abstract graphic composed of the letters 'S' and 'Y' arranged in a grid-like pattern. The overall shape is a stylized 'Y' or a complex letter 'S'. The top part is a wide horizontal bar made of 'S's, with 'Y's forming a central vertical stem. The sides of the 'Y' are also formed by 'S's and 'Y's, creating a complex, layered structure. The letters are black on a white background, and the arrangement is highly regular and symmetrical.

(2)	103
(3)	293
(4)	513
(5)	568
(6)	649
(7)	740
(8)	1092
(9)	1170
(10)	1263
(11)	1422
(12)	1595

DECLARATIONS

SYSGETLKI - GETLKI get lock manager information system service
 GET_REMLKI - Get remote LKI block
 CHECKITEM - Validate item identifier
 MOVEIT - Move data to user's buffer
 SPECIAL - Handle special conditions
 GETLKB - Get specified Lock Block
 VERIFYLOCKID - Verify lock id
 LKISSEARCH_BLOCKING - Search for locks blocking the current lock
 LKISSEARCH_BLOCKEDBY - Search for locks blockedby the current lock
 LKI_ALLOCATE - Allocate a system buffer


```
0000 1 .TITLE SYSGETLKI - GET LOCK MANAGER INFORMATION SYSTEM SERVICE
0000 2 .IDENT 'V04-000'
0000 3
0000 4 *****
0000 5
0000 6 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 * ALL RIGHTS RESERVED.
0000 9
0000 10 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 * TRANSFERRED.
0000 16
0000 17 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 * CORPORATION.
0000 20
0000 21 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23
0000 24 *****
0000 25
0000 26
0000 27 ++
0000 28 FACILITY: VMS Executive, System services.
0000 29
0000 30 ABSTRACT:
0000 31
0000 32 Return system/cluster lock manager information.
0000 33
0000 34 ENVIRONMENT: Kernel Mode
0000 35
0000 36 AUTHOR: Rod N. Gamache, CREATION DATE: 15-November-1982
0000 37
0000 38 MODIFIED BY:
0000 39
0000 40 V03-014 RNG0014 Rod N. Gamache 3-Aug-1984
0000 41 Make all Lock waiting states map to LKISC_WAITING.
0000 42
0000 43 V03-013 RNG0013 Rod N. Gamache 24-Jul-1984
0000 44 Stall access to lock database if cluster is re-configuring,
0000 45 call lock manager routine to perform stall operation.
0000 46
0000 47 V03-012 RNG0012 Rod N. Gamache 01-May-1984
0000 48 Restore the PCB address on successive loops through
0000 49 the main process code, when doing a wildcard search.
0000 50
0000 51 V03-011 RNG0011 Rod N. Gamache 26-Mar-1984
0000 52 Fix invalid REMLKID that is returned on Local copy LOCKS.
0000 53
0000 54 V03-010 RNG0010 Rod N. Gamache 21-Mar-1984
0000 55 Return correct EPID value, return 2 more longwords in the list
0000 56 items (REMLKID & remCSID). Set size of individual items in
0000 57 list requests.
```

```
0000 58 : Return SS$_IVMODE on access mode violations.
0000 59 :
0000 60 : V03-009 CWH3009 CW Hobbs 28-Feb-1984
0000 61 : Change IPL synchronization so that $GETLKI can be called
0000 62 : at IPL <= IPL$_ASTDEL. This lets $GETDVI interrogate
0000 63 : the XQP's lock value block so that $GETDVI can return
0000 64 : the correct value for DVI$_FREEBLOCKS.
0000 65 :
0000 66 : V03-008 RNG0008 Rod N. Gamache 05-Dec-1983
0000 67 : Change references to LOCK STRUCTURES to reflect changes made
0000 68 : in the Lock Manager.
0000 69 :
0000 70 : V03-007 RNG0007 Rod N. Gamache 07-Oct-1983
0000 71 : Fix synchronization problem caused by exec routine that
0000 72 : lowers IPL; wrote inline code to replace exec routine.
0000 73 :
0000 74 : V03-006 CWH3006 CW Hobbs 23-Sep-1983
0000 75 : Fix broken branch
0000 76 :
0000 77 : V03-005 RNG0005 Rod N. Gamache 31-Aug-1983
0000 78 : Deliver AST's only on success.
0000 79 : Allow EXEC mode and KERNEL mode users access to system locks.
0000 80 : Return zero REMLKID if CSID is zero.
0000 81 :
0000 82 : V03-004 RNG0004 Rod N. Gamache 05-Aug-1983
0000 83 : Add REMLKID item code.
0000 84 : Return SS$_NOMORELOCK error instead of SS$_NOMOREPROC.
0000 85 : Add support for distributed list items (LOCKS, BLOCKEDBY
0000 86 : and BLOCKING).
0000 87 : Make sure user has sufficient BYCNT quota for list operations.
0000 88 : Return proper CSID in the event the CSID of the RSB is zero.
0000 89 :
0000 90 : V03-003 RNG0003 Rod N. Gamache 05-May-1983
0000 91 : Return "external" PID wherever necessary. Return
0000 92 : SS$_NOWORLD error instead of SS$_NOPRIV.
0000 93 : Add support for distributed GETLKI.
0000 94 :
0000 95 : V03-002 SRB0073 Steve Beckhardt 30-Mar-1983
0000 96 : Fix broken ASSUME statement.
0000 97 :
0000 98 : V03-001 RNG0001 Rod N. Gamache 14-Mar-1983
0000 99 : Remove SYSNAM bit from RMOD field. Change RMOD to be a
0000 100 : full byte. Use RMOD in RSB rather than LKB.
0000 101 :--
```



```
0000 160
0000 161 :
0000 162 : EQUATED SYMBOLS:
0000 163 :
0000 164 :
00000002 0000 165 MAXSTRUC = 2 ; Maximum number of structures
0000 166 :
00000004 0000 167 EFN = 4 ; event flag number argument
00000008 0000 168 LKID = 8 ; address of the lock ID
0000000C 0000 169 ITMLST = 12 ; address of item identifiers
00000010 0000 170 IOSB = 16 ; I/O status block address
00000014 0000 171 ASTADR = 20 ; ast routine address
00000018 0000 172 ASTPRM = 24 ; ast parameter
0000001C 0000 173 RESERV = 28 ; RESERVED
0000 174 :
0000 175 :
0000 176 : One quadword local is left on stack for routines which may
0000 177 : manipulate values before returning them.
0000 178 :
0000 179 :
FFFFFFFB 0000 180 LOCAL_SPACE = -8
FFFFFFFC 0000 181 SAVED_IPL = -4 ; We will reference stored IPL off the frame
0000 182 :
00000005 0000 183 MAX_LKB_ITEM = <LKIS_LASTLKB*-XFF>-1 ; maximum LKBTBL item number
0000000B 0000 184 MAX_RSB_ITEM = <LKIS_LASTRSB*-XFF>-1 ; maximum RSBTBL item number
0000 185 :
0000 186 :
0000 187 : Data type codes (all numeric types have same code)
0000 188 :
0000 189 :
00000000 0000 190 VALUE = 0 ; numeric value
00000001 0000 191 BSTRING = 1 ; blank filled string
00000002 0000 192 CSTRING = 2 ; counted ascii string
0000 193 :
0000 194 : AST control block extensions
0000 195 :
0000 196 SDEFINI ACB
0000 197 :
0000001C 0000 198 . = ACBSL_EAST+4 ;
0000 199 :
001C 200 SDEF ACB_L_DADDR .BLKL 1 ; data buffer address
0020 201 SDEF ACB_L_EFN .BLKL 1 ; event flag number
0024 202 SDEF ACB_L_IOSB .BLKL 1 ; completion AST routine addr
0028 203 SDEF ACB_L_OPID .BLKL 1 ; original requester's PID
002C 204 SDEF ACB_L_COUNT .BLKL 1 ; item descriptor count
0030 205 SDEF ACB_L_ILIST ; item descriptor list
0030 206 :
0000000C 0030 207 ACB_C_IDESC = 12 ; item descriptor size
0030 208 :
0030 209 SDEFEND ACB
0000 210 :
0000 211 :
0000 212 :
0000 213 : OWN STORAGE:
0000 214 :
0000 215 :
00000000 216 .PSECT WSYSGETLKI ; Non-paged PSECT
```

```
0000 217
0000 218
0000 219
0000 220 : This array contains the maximum item number for both of the item
0000 221 : data structures, indexed by structure number.
0000 222
0000 223
0000 224 MAXCOUNT:
05 0000 225 : .BYTE MAX_LKB_ITEM
08 0001 226 : .BYTE MAX_RSB_ITEM
0002 227
0002 228
0002 229 : The tables contain a word offset followed by a byte code for each item
0002 230 : followed by a byte of structure type. The code contains the length of
0002 231 : the item in the low five bits, and the item type in the high three bits.
0002 232 : The types are value, counted string, and blank filled string.
0002 233
0002 234
0002 235 LKBTBL:
00000026 0002 236 : .BLKB 6*<MAX_LKB_ITEM+1> ; define LKB table
0026 237 RSBTBL:
0000005C 0026 238 : .BLKB 6*<MAX_RSB_ITEM+1> ; define RSB table
005C 239 : .SAVE ; save current location counter
005C 240
005C 241
005C 242 : Define entries to LKBTBL
005C 243
00000000 005C 244 : LMSGBK_ZERO = 0 ; Define empty holder
005C 245
005C 246 LKBITH PID,L EPID,VALUE,4 ; EPID of owner process
0008 247 LKBITH LCKREFCNT,W REFCNT,VALUE,2 ; sub-lock reference count
001A 248 LKBITH STATE,B RMODE,VALUE,3,L_STATE ; current state of lock
000E 249 LKBITH PARENT,C_PARENT,VALUE,4 ; LOCK ID of parent lock
0014 250 LKBITH LOCKID,L_LKID,VALUE,4 ; LOCK ID of lock
0020 251 LKBITH REMLKID,C_REMLKID,VALUE,4 ; Remote LOCK ID of lock
0026 252
0026 253
0026 254 : Define entries to RSBTBL
0026 255
0026 256
0026 257
0026 258 RSBITH RESNAM,B RSHLEN,CSTRING,31 ; resource name
0032 259 RSBITH RSBREFCNT,W REFCNT,VALUE,2,L_RSBREFCNT ; sub-resource reference count
0038 260 RSBITH VALBLK,Q_VALBLK,BSTRING,16,Q_VALBLK ; value block
003E 261 RSBITH SYSTEM,L_CSID,VALUE,4 ; system id of system which has
0044 262 : the master copy of resource
0044 263 RSBITH NAMESPACE,W_GROUP,VALUE,4 ; resource name space
002C 264 RSBITH LCKCOUNT,L_GRPFL,VALUE,4,L_LCKCOUNT ; count of locks granted on reso
004A 265 RSBITH BLOCKEDBY,C_GRPFL,VALUE,- ; list of locks blocked by LKID
004A 266 LKISC_LENGTH
0050 267 RSBITH BLOCKING,L_WTOFL,VALUE,- ; list of locks blocking LKID
0050 268 LKISC_LENGTH
0056 269 RSBITH LOCKS,L_GRPFL,VALUE,LKISC_LENGTH ; list of associated locks
005C 270
005C 271
0000005C 272 : .RESTORE ; restore location counter
005C 273
```



```
005C 274 :  
005C 275 : Table to define items which must be handled  
005C 276 : by action routines.  
005C 277 :  
005C 278 :  
005C 279 SPECIAL:  
005C 280 SPECIAL_ITEM PID,SPC_PID : PID of owner process  
0062 281 SPECIAL_ITEM STATE,SPC_STATE : current state of lock  
0068 282 SPECIAL_ITEM PARENT,SPC_PARENT : LOCK ID of parent lock  
006E 283 SPECIAL_ITEM SYSTEM,SPC_SYSTEM : CSID of master  
0074 284 SPECIAL_ITEM NAMESPACE,SPC_NAMESPACE : resource name space  
007A 285 SPECIAL_ITEM LCKCOUNT,SPC_LCKCOUNT : count of locks granted on resource  
0080 286 SPECIAL_ITEM BLOCKEDBY,SPC_BLOCKEDBY : list of locks blocked by LKID  
0086 287 SPECIAL_ITEM BLOCKING,SPC_BLOCKING : list of locks blocking LKID  
008C 288 SPECIAL_ITEM LOCKS,SPC_LOCKS : list of associated locks  
0092 289 SPECIAL_ITEM REMLKID,SPC_REMLKID : Remote lock id  
0098 290  
0000000A 0098 291 SPECIAL_LEN = <.-SPECIAL>/6 : compute number of entries
```



```
0098 293 .SBTTL SYSGETLKI - GETLKI get lock manager information system service
0098 294
0098 295 ++
0098 296
0098 297 FUNCTIONAL DESCRIPTION:
0098 298
0098 299 This service allows a process to receive information about the
0098 300 locks, or any process locks which it has the privilege to examine.
0098 301
0098 302 CALLING SEQUENCE:
0098 303
0098 304 CALLS/CALLG
0098 305
0098 306 Actually, this routine MUST be called through the system
0098 307 service dispatcher.
0098 308
0098 309 INPUTS:
0098 310
0098 311 R4 PCB address of requesting process
0098 312
0098 313 EFN(AP) number of the event flag to set when all of the
0098 314 requested data is valid.
0098 315 LKID(AP) address of a longword containing the process ID of the
0098 316 process for which the information is being requested
0098 317 ITMLST(AP) address of a list of item descriptors of the form:
0098 318
0098 319 +-----+
0098 320 | ITEM CODE | BUF. LENGTH |
0098 321 +-----+
0098 322 | BUFFER ADDRESS |
0098 323 +-----+
0098 324 | ADDRESS TO RETURN LENGTH |
0098 325 +-----+
0098 326
0098 327 IOSB(AP) address of a quadword I/O status block to receive final
0098 328 status
0098 329 ASTADR(AP) address of an AST routine to be called when all of the
0098 330 requested data has been supplied.
0098 331 ASTPRM(AP) 32 bit ast parameter
0098 332
0098 333 IMPLICIT INPUTS:
0098 334
0098 335 IPL <= IPL$ASTDEL This allows other system services which are
0098 336 holding mutexes to call $GETLKI.
0098 337
0098 338 OUTPUTS:
0098 339
0098 340 none
0098 341
0098 342 IMPLICIT OUTPUTS:
0098 343
0098 344 none
0098 345
0098 346 ROUTINE VALUE:
0098 347
0098 348 SS$NORMAL normal completion.
0098 349 SS$ACCVIO ITMLST can not be read by the calling access mode.
```

```
0098 350 : or the return buffer or return length word can not
0098 351 : be written by the calling access mode.
0098 352 : an invalid item identifier was supplied.
0098 353 : lock id specified is not valid.
0098 354 : data has overflowed the user buffer.
0098 355 : SYSLOCK privilege is needed to access this information.
0098 356 : WORLD privilege is needed to access this information.
0098 357 : User's AST or BYTLM quota has been exceeded.
0098 358 : Insufficient non-paged dynamic memory.
0098 359 : No more locks on "wildcard" search operation.
0098 360 :
0098 361 : SIDE EFFECTS:
0098 362 :
0098 363 : none
0098 364 :
0098 365 :
0000 0000 .PSECT YEXEPAGED ; Only entry mask in this program section
0000 0000 .ENTRY EXESGETLK1,"M(R2,R3,R4,R5,R6,R7,R8,R9,R10,R11)"
0000 0000 JMP EXE_GETLK1 ; Transfer to real procedure
0000 0000 .PSECT WSYSGETLK1
0098 371 EXE_GETLK1:
0098 372 DSBINT @IPLS SYNCH ; Raise IPL to check lock mgr database
0098 373 JSB G-LCKSCHECK_STALL ; ... stall if needed (in CALLER's mode)
0098 374 SETIPL @IPLS_ASTDEL ; Set IPL to ASTDEL
0098 375 MOVAL LOCAL_SPACE(SP),SP ; Allocate local space on stack
0098 376 CLRL R11 ; Assume no remote LOCK information
0098 377 2B: MOVL SCHSGL_CURPCB,R4 ; Reset PCB address
0098 378 MOVL R11,R0 ; Any remote LOCK BLOCK?
0098 379 BEQL 3B ; Br if not, okay
0098 380 CLRL R11 ; No more remote lock block
0098 381 JSB EXESDEANONPAGED ; Else, deallocate the remote lock block
0098 382 3B: BSBW GETLK0 ; Get LKB address of desired lock
0098 383 BLBC R0,178 ; Exit if invalid LKID specified
0098 384 :
0098 385 : Check for, and clear event flag
0098 386 :
0098 387 :
0098 388 :
0098 389 MOVZBL EFN(AP),R3 ; Get event flag number
0098 390 JSB SCHSCLREF ; Clear this event flag
0098 391 BLBC R0,178 ; And return on errors.
0098 392 :
0098 393 : Check for, and clear possible IOSB
0098 394 :
0098 395 :
0098 396 :
0098 397 :
0098 398 :
0098 399 :
0098 400 :
0098 401 : Validate AST, if present. Note R4 still has our PCB address, and R9
0098 402 : has the LKB address of the lock we want information from.
0098 403 :
0098 404 :
0098 405 :
0098 406 5B: TSTL ASTADR(AP) ; No AST to check.
0098 407 BEQL 7B ; Is quota exceeded?
0098 408 TSTW PCBSW_ASTCNT(R4)
```



```
7B 15 00ED 407 BLEQ 358 ; Branch if so and return error
      00EF 408
      00EF 409 ; Check if information is contained on another system in the cluster
      00EF 410
      00EF 411 7B: BSBW GET REMLKI ; Get remote LKI block if needed
      2E 50 E9 412 BLBC R0,T78 ; Exit on error
      00EF 413
      00EF 414 ; Loop through the item descriptor blocks, validating the requested item
      00EF 415 ; identifiers and moving accessible items. A zero item identifier terminates
      00EF 416 ; the list.
      00EF 417
      00EF 418 ; At this point:
      00EF 419
      00EF 420 ; R4 = PCB address
      00EF 421 ; R9 = LKB address
      00EF 422 ; R11 = Remote lock block information or zero
      00EF 423 ; AP = Pointer to argument list
      00EF 424
      55 0C AC D0 425 10B: MOVL ITHLST(AP),R5 ; Get item descriptor list address
      56 85 3C 426 IFNORD #4,(R5),308 ; Check first longword readable
      51 85 3C 427 15B: MOVZWL (R5)+,R6 ; Get buffer size
      6D 13 0102 428 MOVZWL (R5)+,R1 ; Get item identifier
      0103 429 BEQL 508 ; Done if zero, take normal exit
      0104 430 IFNORD #12,(R5),308 ; Check rest of this descriptor ...
      57 85 7D 431 ; ... plus first longword of next one
      51 51 D0 432 MOVQ (R5)+,R7 ; Get buffer address and return address
      50 57 D0 433 PUSHL R1 ; Save R1 across accessibility check
      51 56 D0 434 MOVL R7,R0 ; Buffer address to R0
      53 53 D4 435 MOVL R6,R1 ; And size to R1
      00000000'EF 16 0118 436 CLRL R3 ; PROBE will use PSL<PRVMOD>
      51 51 BED0 437 JSB EXESPROBEW ; Check write accessibility of buffer
      51 50 E9 438 POPL R1 ; Restore R1 for use by CHECKITEM
      17B: 439 BLBC R0,GRET ; Return error if inaccessible
      440
      441 ; We will raise IPL to IPLS SYNCH to lock down the LKB. We will
      442 ; have to verify that the LKB is still valid, before proceeding.
      443
      444 ; The IPL will be restored by the MOVEIT routine just before copying
      445 ; the data to the user's buffer. This is done to allow the SPC_XXX
      446 ; routines to gather up any additional information that needs to be
      447 ; returned to the user, without verifying that the LKB address is
      448 ; still valid.
      449
      450 ; SETIPL #IPLS_SYNCH ; Raise IPL to sync access to structures
      451 ; ; can't reference user's process space
      452
      453 MOVZWL LKBSL LKID(R9),R4 ; Get lock index
      454 CMPL R4,LCKBGL_MAXID ; Is the lock index still ok?
      455 BGTNU 208 ; Or if no, check for error condition
      456 CMPL R9,BLCKBGL_IDTBL[R4] ; Is the lock address still the same?
      457 BEQL 258 ; Or if yes, okay to proceed
      458 20B: SETIPL #IPLS_ASTDEL ; Restore the IPL on error condition
      459 TSTL BLKIDTAP ; Is this a "wildcard" search?
      460 BGTN 258 ; Or if no, continue
      461 ARU 28 ; Else, try for next lock
      462 23B: MOVZWL #SSB_IVLOCKID,R0 ; Invalid lock id
      463 BRG GRET ; Return to user
      464
      465 ;
```



```
0152 464 ; Check item code and return the info to user.
0153 465
0154 466 258: PUSHL R5 ; Save R5 from action routines
0155 467 BSBW CHECKITEM ; Validate identifier and get item info.
0156 468 BLBC R0,408 ; Invalid item if error
0157 469 BSBW MOVEIT ; Move item to user
0158 470 ; NOTE: IPL is restored to IPLS_ASTDEL
0159 471 POPL R5 ; Restore R5
0160 472 BLBS R0,158 ; Back for next descriptor if ok
0161 473 BRB GREY ; Else, return error
0162 474
0163 475 308: MOVZWL #SSB_ACCVIO,R0 ; Access violation
0164 476 BRB GREY
0165 477
0166 478 358: MOVZWL #SSB_EXQUOTA,R0 ; AST quota exceeded
0167 479 BRB GREY
0168 480
0169 481 408: MOVZWL #SSB_BADPARAM,R0 ; Illegal item or request
0170 482 BRB GREY
0171 483
0172 484 508: MOVZWL #SSB_NORMAL,R0 ; Normal return
0173 485
0174 486 ; Set the event flag, post the completion status, and declare a completion AST
0175 487 GREY:
0176 488
0177 489 PUSHL R0 ; Save completion status
0178 490 MOVL R11,R0 ; Any remote lock block?
0179 491 BEQL SB ; Or if not, okay
0180 492 JSB EXESDEANONPAGED ; Else, deallocate the remote lock block
0181 493 SETIPL SAVED_IPL(FP) ; Restore IPL to that on entry to service
0182 494 MOVL SCHSGC_CURPCB,R4 ; Get PCB address
0183 495 MOVL PCBSL_PID(R4),R1 ; Get process's PID
0184 496 CLRL R2 ; Set null priority increment
0185 497 MOVL EFN(AP),R3 ; Get event flag number to set
0186 498 JSB SCHSPOSTEF ; Set the event flag
0187 499 MOVL IOSB(AP),R1 ; Get address of IOSB
0188 500 BEQL 208 ; Branch if none
0189 501 IFNOWRT #B(R1),208 ; Check if writable
0190 502 MOVL (SP),R1 ; Store completion status
0191 503 MOVL ASTADR(AP),R5 ; Get address of AST routine
0192 504 BEQL 308 ; Branch if none specified
0193 505 BLBC (SP),308 ; No completion AST on error!
0194 506 MOVPSL R4 ; Get PSL
0195 507 EXTRV #PSLSV_PRIVMOD,#PSLSS_PRIVMOD,R4,R4 ; Extract previous mode
0196 508 SDCLAST_S(R5),ASTPRM(AP),R4 ; Queue the completion AST
0197 509 308: POPL R0 ; Restore completion status
0198 510 RET ; And return.
0199 511
```

55 DD 0152 464
009C 50 0153 465
15 50 0154 466
00FD 50 0155 467
55 BEDD 0156 468
9C 50 0157 469
12 11 0158 470
50 0C 3C 0159 471
0D 11 0160 472
50 1C 3C 0161 473
08 11 0162 474
50 14 3C 0163 475
03 11 0164 476
50 01 3C 0165 477
50 50 DD 0166 478
50 58 DD 0167 479
06 13 0168 480
00000000 EF 16 0169 481
54 00000000 EF DD 0170 482
51 60 A4 DD 0171 483
52 04 B4 DD 0172 484
53 04 AC DD 0173 485
00000000 EF 16 0174 486
51 10 AC DD 0175 487
09 13 0176 488
61 6E DD 0177 489
55 14 AC DD 0178 490
18 13 0179 491
15 6E EF 0180 492
54 DC 0181 493
54 54 02 16 EF 0182 494
50 BEDD 0183 495
04 0184 496
0185 497
0186 498
0187 499
0188 500
0189 501
018A 502
018B 503
018C 504
018D 505
018E 506
018F 507
0190 508
0191 509
0192 510
0193 511

```
01D0 513 .SBTTL GET_REPLKI- Get remote LKI block
01D0 514
01D0 515
01D0 516
01D0 517
01D0 518
01D0 519
01D0 520
01D0 521
01D0 522
01D0 523
01D0 524
01D0 525
01D0 526
01D0 527
01D0 528
01D0 529
01D0 530
01D0 531
01D0 532
01D0 533
01D0 534
01D0 535
01D0 536
01D0 537
01D0 538
01D0 539
01D0 540
01D0 541
01D0 542
01D0 543
01D0 544
01D0 545
01D0 546
01D0 547
01D0 548
01D0 549
01D0 550
01D0 551
01D0 552
01D0 553
01D0 554
01D0 555
01D0 556
01D0 557
01D0 558
01D0 559
01D0 560
01D0 561
01D0 562
01D0 563
01D0 564
01D0 565
01D0 566

**
FUNCTIONAL DESCRIPTION:
    Routine to get the remote LKI block if necessary.
CALLING SEQUENCE:
    JSB/BSB
INPUTS:
    R4      PCB address
    R9      LKB address
    R11     ZERO
IMPLICIT INPUTS:
    IPL = IPLS_ASTDEL
OUTPUTS:
    R0      success/failure of operation + special flags
    R4      PCB address
    R9      LKB address
    R11     Address of remote LKI block or zero
IMPLICIT OUTPUTS:
    none
SIDE EFFECTS:
    R0-R3,R8 destroyed.
--
    .ENABL  LSB
GET_REPLKI:
    MOVB    #1,R0
    DBS     @LKBSV_RSTCPY -
            LKBSW_STATUS(R9),108
    MOVL    LKBSL_RSB(R9),R8
    MOVL    RBSBL_CSID(R8),R3
    BEQL    108
    SETIPL  @IPLS_SYNCH
    JSB     @LKBSNO_STDREQ
    MOVL    SCHBGL_CURPCB,R4
    RSB
    108:
    .DSABL  LSB

: Get remote LKI block
: Assume success
: Br if this is the master copy,
:   information is local to this system
: Get RSB address
: Is this a process copy?
: Br if not, information is still local
: Raise IPL to SYNCH
: And send request for information
:   to remote system
: Get our PCB address
: Return to caller
```

```
50 01 90
   04 E0
1A 2A A9
58 50 A9 D0
53 38 A8 D0
   10 13
00000000*GF 16
54 00000000*EF D0
   05
```

```
01F3 568 .SBTTL CHECKITEM - Validate item identifier
01F3 569
01F3 570
01F3 571
01F3 572
01F3 573
01F3 574
01F3 575
01F3 576
01F3 577
01F3 578
01F3 579
01F3 580
01F3 581
01F3 582
01F3 583
01F3 584
01F3 585
01F3 586
01F3 587
01F3 588
01F3 589
01F3 590
01F3 591
01F3 592
01F3 593
01F3 594
01F3 595
01F3 596
01F3 597
01F3 598
01F3 599
01F3 600
01F3 601
01F3 602
01F3 603
01F3 604
01F3 605
01F3 606
01F3 607
01F3 608
01F3 609
01F3 610
01F3 611
01F3 612
01F3 613
01F3 614
01F3 615
01F3 616
01F3 617
01F3 618
01F3 619
01F3 620
01F3 621
01F3 622
01F3 623
01F3 624

52 51 53 50 04 01F3 610 CLRL R0 ; Assume bad item code
08 08 08 51 9A 01F3 611 MOVZBL R1,R3 ; Get item number
02 52 91 13 01F3 612 EXTZV #8,#8,R1,R2 ; Get structure number
FDF5 CF42 55 1A 0202 613 BEQL 80$ ; Error if structure number zero
53 91 0204 614 CMPB R2,#MAXSTRUC ; Structure number valid?
4D 1A 020A 615 BGTRU 80$ ; Error if not
020C 616 CMPB R3,MAXCOUNT-1[R2] ; Check max item values (1 origin)
0214 617 BGTRU 80$ ; Error if illegal item number
0214 618 CASE R2,<10$,30$>B,#1 ; Case on structure base
0214 619 ; LKB return item
0214 620
0214 621
0214 622 10$: NOVL R9,R4 ; Get back LKB address
55 FDE7 CF DE 0217 623 NOVAL LKBTL,R5 ; Get address of LKB item table
09 11 021C 624 BRB 40$ ; Continue
```



```

      54 50 A9 D0 021E 625
55 55 FE00 CF DE 021F 626
      53 53 01 70 0220 627
      53 6543 DE 0221 628
      53 50 CO 0222 629
      53 50 CO 0223 630
      54 50 CO 0224 631
      50 01 CO 0225 632
      02 A3 CO 0226 633
      54 02 A3 CO 0227 634
      00 54 50 01 CO 0228 635
55 63 03 05 CO 0229 636
55 63 03 00 CO 0230 637
      03 CO 0231 638
      03 CO 0232 639
      03 CO 0233 640
      03 CO 0234 641
      03 CO 0235 642
      03 CO 0236 643
      03 CO 0237 644
      03 CO 0238 645
      03 CO 0239 646
      03 CO 023A 647

      RSB return item
30B: MOVL LKBSL,RSB(R9),R4      : Get resource block address
      MOVAL RSBTBC,R5           : Get address of PHD item table
40B: ASHL #1,R3,R0              : Double item number
      MOVAL (R5)(R3),R3         : Compute address in item table
      ADDL R0,R3                : Get offset into data structure
      MOVZWL (R5)+,R5           : Form complete address
      ADDL R5,R4                : Set successful return
      MOVL #1,R0                : Is there a remote LKI block?
      TSTL R11                  : Br if not, continue
      BEQL 50B                  : Is this item in remote LKI block?
      TSTM 2(R3)                : Br if not
      BEQL 50B                  : Else, get offset in remote LKI block
      MOVZWL 2(R3),R4           : Form complete address
      ADDL R11,R4               : Indicate that no special lookup needed
      DBSS #1,R0,50B            : Get item type code
30B: EXTZV #5,R3,(R3),R5       : Get item length
      EXTZV #0,R5,(R3),R3      : Return to caller
80B: RSB
```

```
02SA 649 .SBTTL MOVEIT - Move data to user's buffer
02SA 650
02SA 651
02SA 652
02SA 653
02SA 654
02SA 655
02SA 656
02SA 657
02SA 658
02SA 659
02SA 660
02SA 661
02SA 662
02SA 663
02SA 664
02SA 665
02SA 666
02SA 667
02SA 668
02SA 669
02SA 670
02SA 671
02SA 672
02SA 673
02SA 674
02SA 675
02SA 676
02SA 677
02SA 678
02SA 679
02SA 680
02SA 681
02SA 682
02SA 683
02SA 684
02SA 685
02SA 686
02SA 687
02SA 688
02SA 689
02SA 690
02SA 691
02SA 692
02SA 693
02SA 694
02SA 695
02SA 696
02SA 697
02SA 698
02SA 699
02SA 700
02SA 701
02SA 702
02SA 703
02SA 704
02SA 705
02SC 703
0260 704
0262 705

02 50 SA D4
01 EQ
4B 10

        .SBTTL MOVEIT - Move data to user's buffer
        **
        FUNCTIONAL DESCRIPTION:
            Move the requested data to user buffer. Zero fill to end of buffer.
            Return actual data length to user. Assumes user's buffer has
            been probed.
        CALLING SEQUENCE:
            JSB/BSB
        INPUTS:
            R0      special lookup flag
            R1      item identifier
            R2      data structure number
            R3      item length
            R4      item address
            R5      item type code
            R6      user buffer length
            R7      user buffer address
            R8      address to return length
            R9      LKB address
        IMPLICIT INPUTS:
            IPL = IPLB_SYNCH
        OUTPUTS:
            none
        IMPLICIT OUTPUTS:
            IPL = IPLB_ASTDEL
        ROUTINE VALUE:
            SSB_NORMAL      Normal successful completion
            SSB_ACCVIO      Access violation on attempt to access return size
        SIDE EFFECTS:
            Registers R1-R4 destroyed
        MOVEIT:
            Call routine to check for special conditions
            CLRL R10
            BBS #1,R0,SB
            BSRB CHECK_SPC
            SETIPL #IPLB_ASTDEL
            ; No buffer to deallocate - yet!
            ; Or if no special lookup needed
            ; Check for special actions
            ; Restore IPL to ASTDEL
```

```
2E 50 E9 0265 706 BLBC R0,408 ; Br if error
0266 707
0267 708
0268 709
0269 710
55 02 D1 026A 711 ; Check for counted string, and find actual length if so.
03 12 026B 712
53 84 9A 026C 713 ; Is this special string?
026D 714 ; Br if not
026E 715 ; Get length and skip length byte
026F 716
0270 717
0271 718
0272 719
0273 720
0274 721
0275 722
0276 723
0277 724
0278 725
0279 726
027A 727
027B 728
027C 729
027D 730
027E 731
027F 732
0280 733
0281 734
0282 735
0283 736
0284 737
0285 738
0286 739
0287 740
0288 741
0289 742
028A 743
028B 744
028C 745
028D 746
028E 747
028F 748
0290 749
0291 750
0292 751
0293 752
0294 753
0295 754
0296 755
0297 756
0298 757
0299 758
029A 759
029B 760
029C 761
029D 762
029E 763
029F 764
02A0 765
02A1 766
02A2 767
02A3 768
02A4 769
02A5 770
02A6 771
02A7 772
02A8 773
02A9 774
02AA 775
02AB 776
02AC 777
02AD 778
02AE 779
02AF 780
02B0 781
02B1 782
02B2 783
02B3 784
02B4 785
02B5 786
02B6 787
02B7 788
02B8 789
02B9 790
02BA 791
02BB 792
02BC 793
02BD 794
02BE 795
02BF 796
02C0 797
02C1 798
02C2 799
02C3 800
02C4 801
02C5 802
02C6 803
02C7 804
02C8 805
02C9 806
02CA 807
02CB 808
02CC 809
02CD 810
02CE 811
02CF 812
02D0 813
02D1 814
02D2 815
02D3 816
02D4 817
02D5 818
02D6 819
02D7 820
02D8 821
02D9 822
02DA 823
02DB 824
02DC 825
02DD 826
02DE 827
02DF 828
02E0 829
02E1 830
02E2 831
02E3 832
02E4 833
02E5 834
02E6 835
02E7 836
02E8 837
02E9 838
02EA 839
02EB 840
02EC 841
02ED 842
02EE 843
02EF 844
02F0 845
02F1 846
02F2 847
02F3 848
02F4 849
02F5 850
02F6 851
02F7 852
02F8 853
02F9 854
02FA 855
02FB 856
02FC 857
02FD 858
02FE 859
02FF 860
0300 861
0301 862
0302 863
0303 864
0304 865
0305 866
0306 867
0307 868
0308 869
0309 870
030A 871
030B 872
030C 873
030D 874
030E 875
030F 876
0310 877
0311 878
0312 879
0313 880
0314 881
0315 882
0316 883
0317 884
0318 885
0319 886
031A 887
031B 888
031C 889
031D 890
031E 891
031F 892
0320 893
0321 894
0322 895
0323 896
0324 897
0325 898
0326 899
0327 900
0328 901
0329 902
032A 903
032B 904
032C 905
032D 906
032E 907
032F 908
0330 909
0331 910
0332 911
0333 912
0334 913
0335 914
0336 915
0337 916
0338 917
0339 918
033A 919
033B 920
033C 921
033D 922
033E 923
033F 924
0340 925
0341 926
0342 927
0343 928
0344 929
0345 930
0346 931
0347 932
0348 933
0349 934
034A 935
034B 936
034C 937
034D 938
034E 939
034F 940
0350 941
0351 942
0352 943
0353 944
0354 945
0355 946
0356 947
0357 948
0358 949
0359 950
035A 951
035B 952
035C 953
035D 954
035E 955
035F 956
0360 957
0361 958
0362 959
0363 960
0364 961
0365 962
0366 963
0367 964
0368 965
0369 966
036A 967
036B 968
036C 969
036D 970
036E 971
036F 972
0370 973
0371 974
0372 975
0373 976
0374 977
0375 978
0376 979
0377 980
0378 981
0379 982
037A 983
037B 984
037C 985
037D 986
037E 987
037F 988
0380 989
0381 990
0382 991
0383 992
0384 993
0385 994
0386 995
0387 996
0388 997
0389 998
038A 999
038B 1000
038C 1001
038D 1002
038E 1003
038F 1004
0390 1005
0391 1006
0392 1007
0393 1008
0394 1009
0395 1010
0396 1011
0397 1012
0398 1013
0399 1014
039A 1015
039B 1016
039C 1017
039D 1018
039E 1019
039F 1020
03A0 1021
03A1 1022
03A2 1023
03A3 1024
03A4 1025
03A5 1026
03A6 1027
03A7 1028
03A8 1029
03A9 1030
03AA 1031
03AB 1032
03AC 1033
03AD 1034
03AE 1035
03AF 1036
03B0 1037
03B1 1038
03B2 1039
03B3 1040
03B4 1041
03B5 1042
03B6 1043
03B7 1044
03B8 1045
03B9 1046
03BA 1047
03BB 1048
03BC 1049
03BD 1050
03BE 1051
03BF 1052
03C0 1053
03C1 1054
03C2 1055
03C3 1056
03C4 1057
03C5 1058
03C6 1059
03C7 1060
03C8 1061
03C9 1062
03CA 1063
03CB 1064
03CC 1065
03CD 1066
03CE 1067
03CF 1068
03D0 1069
03D1 1070
03D2 1071
03D3 1072
03D4 1073
03D5 1074
03D6 1075
03D7 1076
03D8 1077
03D9 1078
03DA 1079
03DB 1080
03DC 1081
03DD 1082
03DE 1083
03DF 1084
03E0 1085
03E1 1086
03E2 1087
03E3 1088
03E4 1089
03E5 1090
03E6 1091
03E7 1092
03E8 1093
03E9 1094
03EA 1095
03EB 1096
03EC 1097
03ED 1098
03EE 1099
03EF 1100
03F0 1101
03F1 1102
03F2 1103
03F3 1104
03F4 1105
03F5 1106
03F6 1107
03F7 1108
03F8 1109
03F9 1110
03FA 1111
03FB 1112
03FC 1113
03FD 1114
03FE 1115
03FF 1116
0400 1117
0401 1118
0402 1119
0403 1120
0404 1121
0405 1122
0406 1123
0407 1124
0408 1125
0409 1126
040A 1127
040B 1128
040C 1129
040D 1130
040E 1131
040F 1132
0410 1133
0411 1134
0412 1135
0413 1136
0414 1137
0415 1138
0416 1139
0417 1140
0418 1141
0419 1142
041A 1143
041B 1144
041C 1145
041D 1146
041E 1147
041F 1148
0420 1149
0421 1150
0422 1151
0423 1152
0424 1153
0425 1154
0426 1155
0427 1156
0428 1157
0429 1158
042A 1159
042B 1160
042C 1161
042D 1162
042E 1163
042F 1164
0430 1165
0431 1166
0432 1167
0433 1168
0434 1169
0435 1170
0436 1171
0437 1172
0438 1173
0439 1174
043A 1175
043B 1176
043C 1177
043D 1178
043E 1179
043F 1180
0440 1181
0441 1182
0442 1183
0443 1184
0444 1185
0445 1186
0446 1187
0447 1188
0448 1189
0449 1190
044A 1191
044B 1192
044C 1193
044D 1194
044E 1195
044F 1196
0450 1197
0451 1198
0452 1199
0453 1200
0454 1201
0455 1202
0456 1203
0457 1204
0458 1205
0459 1206
045A 1207
045B 1208
045C 1209
045D 1210
045E 1211
045F 1212
0460 1213
0461 1214
0462 1215
0463 1216
0464 1217
0465 1218
0466 1219
0467 1220
0468 1221
0469 1222
046A 1223
046B 1224
046C 1225
046D 1226
046E 1227
046F 1228
0470 1229
0471 1230
0472 1231
0473 1232
0474 1233
0475 1234
0476 1235
0477 1236
0478 1237
0479 1238
047A 1239
047B 1240
047C 1241
047D 1242
047E 1243
047F 1244
0480 1245
0481 1246
0482 1247
0483 1248
0484 1249
0485 1250
0486 1251
0487 1252
0488 1253
0489 1254
048A 1255
048B 1256
048C 1257
048D 1258
048E 1259
048F 1260
0490 1261
0491 1262
0492 1263
0493 1264
0494 1265
0495 1266
0496 1267
0497 1268
0498 1269
0499 1270
049A 1271
049B 1272
049C 1273
049D 1274
049E 1275
049F 1276
04A0 1277
04A1 1278
04A2 1279
04A3 1280
04A4 1281
04A5 1282
04A6 1283
04A7 1284
04A8 1285
04A9 1286
04AA 1287
04AB 1288
04AC 1289
04AD 1290
04AE 1291
04AF 1292
04B0 1293
04B1 1294
04B2 1295
04B3 1296
04B4 1297
04B5 1298
04B6 1299
04B7 1300
04B8 1301
04B9 1302
04BA 1303
04BB 1304
04BC 1305
04BD 1306
04BE 1307
04BF 1308
04C0 1309
04C1 1310
04C2 1311
04C3 1312
04C4 1313
04C5 1314
04C6 1315
04C7 1316
04C8 1317
04C9 1318
04CA 1319
04CB 1320
04CC 1321
04CD 1322
04CE 1323
04CF 1324
04D0 1325
04D1 1326
04D2 1327
04D3 1328
04D4 1329
04D5 1330
04D6 1331
04D7 1332
04D8 1333
04D9 1334
04DA 1335
04DB 1336
04DC 1337
04DD 1338
04DE 1339
04DF 1340
04E0 1341
04E1 1342
04E2 1343
04E3 1344
04E4 1345
04E5 1346
04E6 1347
04E7 1348
04E8 1349
04E9 1350
04EA 1351
04EB 1352
04EC 1353
04ED 1354
04EE 1355
04EF 1356
04F0 1357
04F1 1358
04F2 1359
04F3 1360
04F4 1361
04F5 1362
04F6 1363
04F7 1364
04F8 1365
04F9 1366
04FA 1367
04FB 1368
04FC 1369
04FD 1370
04FE 1371
04FF 1372
0500 1373
0501 1374
0502 1375
0503 1376
0504 1377
0505 1378
0506 1379
0507 1380
0508 1381
0509 1382
050A 1383
050B 1384
050C 1385
050D 1386
050E 1387
050F 1388
0510 1389
0511 1390
0512 1391
0513 1392
0514 1393
0515 1394
0516 1395
0517 1396
0518 1397
0519 1398
051A 1399
051B 1400
051C 1401
051D 1402
051E 1403
051F 1404
0520 1405
0521 1406
0522 1407
0523 1408
0524 1409
0525 1410
0526 1411
0527 1412
0528 1413
0529 1414
052A 1415
052B 1416
052C 1417
052D 1418
052E 1419
052F 1420
0530 1421
0531 1422
0532 1423
0533 1424
0534 1425
0535 1426
0536 1427
0537 1428
0538 1429
0539 1430
053A 1431
053B 1432
053C 1433
053D 1434
053E 1435
053F 1436
0540 1437
0541 1438
0542 1439
0543 1440
0544 1441
0545 1442
0546 1443
0547 1444
0548 1445
0549 1446
054A 1447
054B 1448
054C 1449
054D 1450
054E 1451
054F 1452
0550 1453
0551 1454
0552 1455
0553 1456
0554 1457
0555 1458
0556 1459
0557 1460
0558 1461
0559 1462
055A 1463
055B 1464
055C 1465
055D 1466
055E 1467
055F 1468
0560 1469
0561 1470
0562 1471
0563 1472
0564 1473
0565 1474
0566 1475
0567 1476
0568 1477
0569 1478
056A 1479
056B 1480
056C 1481
056D 1482
056E 1483
056F 1484
0570 1485
0571 1486
0572 1487
0573 1488
0574 1489
0575 1490
0576 1491
0577 1492
0578 1493
0579 1494
057A 1495
057B 1496
057C 1497
057D 1498
057E 1499
057F 1500
0580 1501
0581 1502
0582 1503
0583 1504
0584 1505
0585 1506
0586 1507
0587 1508
0588 1509
0589 1510
058A 1511
058B 1512
058C 1513
058D 1514
058E 1515
058F 1516
0590 1517
0591 1518
0592 1519
0593 1520
0594 1521
0595 1522
0596 1523
0597 1524
0598 1525
0599 1526
059A 1527
059B 1528
059C 1529
059D 1530
059E 1531
059F 1532
05A0 1533
05A1 1534
05A2 1535
05A3 1536
05A4 1537
05A5 1538
05A6 1539
05A7 1540
05A8 1541
05A9 1542
05AA 1543
05AB 1544
05AC 1545
05AD 1546
05AE 1547
05AF 1548
05B0 1549
05B1 1550
05B2 1551
05B3 1552
05B4 1553
05B5 1554
05B6 1555
05B7 1556
05B8 1557
05B9 1558
05BA 1559
05BB 1560
05BC 1561
05BD 1562
05BE 1563
05BF 1564
05C0 1565
05C1 1566
05C2 1567
05C3 1568
05C4 1569
05C5 1570
05C6 1571
05C7 1572
05C8 1573
05C9 1574
05CA 1575
05CB 1576
05CC 1577
05CD 1578
05CE 1579
05CF 1580
05D0 1581
05D1 1582
05D2 1583
05D3 1584
05D4 1585
05D5 1586
05D6 1587
05D7 1588
05D8 1589
05D9 1590
05DA 1591
05DB 1592
05DC 1593
05DD 1594
05DE 1595
05DF 1596
05E0 1597
05E1 1598
05E2 1599
05E3 1600
05E4 1601
05E5 1602
05E6 1603
05E7 1604
05E8 1605
05E9 1606
05EA 1607
05EB 1608
05EC 1609
05ED 1610
05EE 1611
05EF 1612
05F0 1613
05F1 1614
05F2 1615
05F3 1616
05F4 1617
05F5 1618
05F6 1619
05F7 1620
05F8 1621
05F9 1622
05FA 1623
05FB 1624
05FC 1625
05FD 1626
05FE 1627
05FF 1628
0600 1629
0601 1630
0602 1631
0603 1632
0604 1633
0605 1634
0606 1635
0607 1636
0608 1637
0609 1638
060A 1639
060B 1640
060C 1641
060D 1642
060E 1643
060F 1644
0610 1645
0611 1646
0612 1647
0613 1648
0614 1649
0615 1650
0616 1651
0617 1652
0618 1653
0619 1654
061A 1655
061B 1656
061C 1657
061D 1658
061E 1659
061F 1660
0620 1661
0621 1662
0622 1663
0623 1664
0624 1665
0625 1666
0626 1667
0627 1668
0628 1669
0629 1670
062A 1671
062B 1672
062C 1673
062D 1674
062E 1675
062F 1676
0630 1677
0631 1678
0632 1679
0633 1680
0634 1681
0635 1682
0636 1683
0637 1684
0638 1685
0639 1686
063A 1687
063B 1688
063C 1689
063D 1690
063E 1691
063F 1692
0640 1693
0641 1694
0642 1695
0643 1696
0644 1697
0645 1698
0646 1699
0647 1700
0648 1701
0649 1702
064A 1703
064B 1704
064C 1705
064D 1706
064E 1707
064F 1708
0650 1709
0651 1710
0652 1711
0653 1712
0654 1713
0655 1714
0656 1715
0657 1716
0658 1717
0659 1718
065A 1719
065B 1720
065C 1721
065D 1722
065E 1723
065F 1724
0660 1725
0661 1726
0662 1727
0663 1728
0664 1729
0665 1730
0666 1731
0667 1732
0668 1733
0669 1734
066A 1735
066B 1736
066C 1737
066D 1738
066E 1739
066F 1740
0670 1741
0671 1742
0672 1743
0673 1744
0674 1745
0675 1746
0676 1747
0677 1748
0678 1749
0679 1750
067A 1751
067B 1752
067C 1753
067D 1754
067E 1755
067F 1756
0680 1757
0681 1758
0682 1759
0683 1760
0684 1761
0685 1762
0686 1763
0687 1764
0688 1765
0689 1766
068A 1767
068B 1768
068C 1769
068D 1770
068E 1771
068F 1772
0690 1773
0691 1774
0692 1775
0693 1776
0694 1777
0695 1778
0696 1779
0697 1780
0698 1781
0699 1782
069A 1783
069B 1784
069C 1785
069D 1786
069E 1787
069F 1788
06A0 1789
06A1 1790
06A2 1791
06A3 1792
06A4 1793
06A5 1794
06A6 1795
06A7 1796
06A8 1797
06A9 1798
06AA 1799
06AB 1800
06AC 1801
06AD 1802
06AE 1803
06AF 1804
06B0 1805
06B1 1806
06B2 1807
06B3 1808
06B4 1809
06B5 1810
06B6 1811
06B7 1812
06B8 1813
06B9 1814
06BA 1815
06BB 1816
06BC 1817
06BD 1818
06BE 1819
06BF 1820
06C0 1821
06C1 1822
06C2 1823
06C3 1824
06C4 1825
06C5 1826
06C6 1827
06C7 1828
06C8 1829
06C9 1830
06CA 1831
06CB 1832
06CC 1833
06CD 1834
06CE 1835
06CF 1836
06D0 1837
06D1 1838
06D2 1839
06D3 1840
06D4 1841
06D5 1842
06D6 1843
06D7 1844
06D8 1845
06D9 1846
06DA 1847
06DB 1848
06DC 1849
06DD 1850
06DE 1851
06DF 1852
06E0 1853
06E1 1854
06E2 1855
06E3 1856
06E4 1857
06E5 1858
06E6 1859
06E7 1860
06E8 1861
06E9 1862
06EA 1863
06EB 1864
06EC 1865
06ED 1866
06EE 1867
06EF 1868
06F0 1869
06F1 1870
06F2 1871
06F3 1872
06F4 1873
06F5 1874
06F6 1875
06F7 1876
06F8 1877
06F9 1878
06FA 1879
06FB 1880
06FC 1881
06FD 1882
06FE 1883
06FF 1884
0700 1885
0701 1886
0702 1887
0703 1888
0704 1889
0705 1890
0706 1891
0707 1892
0708 1893
0709 1894
070A 1895
070B 1896
070C 1897
070D 1898
070E 1899
070F 1900
0710 1901
0711 1902
0712 1903
0713 1904
0714 1905
0715 1906
0716 1907
0717 1908
0718 1909
0719 1910
0720 1911
0721 1912
0722 1913
0723 1914
0724 1915
0725 1916
0726 1917
0727 1918
0728 1919
0729 1920
072A 1921
072B 1922
072C 1923
072D 1924
072E 1925
072F 1926
0730 1927
0731 1928
0732 1929
0733 1930
0734 1931
0735 1932
0736 1933
0737 1934
0738 1935
0739 1936
073A 1937
073B 1938
073C 1939
073D 1940
073E 1941
073F 1942
0740 1943
0741 1944
0742 1945
0743 1946
0744 1947
0745 1948
0746 1949
0747 1950
0748 1951
0749 1952
074A 1953
074B 1954
074C 1955
074D 1956
074E 1957
074F 1958
0750 1959
0751 1960
0752 1961
0753 196
```



```
2E 50 E9 0265 706 BLBC R0,40$ ; Br if error
      0268 707 ;
      0268 708 ; Check for counted string, and find actual length if so.
      0268 709 ;
55 02 D1 0268 710 CMPL #CSTRING,R5 ; Is this special string?
      03 12 026B 711 BNEQ 10$ ; Br if not
53 84 9A 026D 712 MOVZBL (R4)+,R3 ; Get length and skip length byte
      0270 713 ;
      0270 714 ; Move the data
      0270 715 ;
67 56 00 64 28 BB 0270 716 10$: PUSHR #M<R3,R5> ; Save registers
      53 2C 0272 717 MOVCS R3,(R4),#0,R6,(R7) ; Move data to user's buffer, zero fill
      28 BA 0278 718 POPR #M<R3,R5> ; Restore registers
      58 D5 027A 719 TSTL R8 ; Did caller want return length?
      15 13 027C 720 BEQL 30$ ; Br if not
      027E 721 IFNOWRT #4,(R8),70$ ; Br if longword not writeable
      56 53 B1 0284 722 CMPW R3,R6 ; See how much was moved
      07 15 0287 723 BLEQ 20$ ; Use valid data length if it fits
      53 56 B0 0289 724 MOVW R6,R3 ; Else give him 'too short' buffer size
00 53 1F E2 028C 725 BBSS #31,R3,20$ ; And return buffer overflow indicator
      68 53 D0 0290 726 20$: MOVL R3,(R8) ; Return length to user
      50 01 9A 0293 727 30$: MOVZBL S#SS$_NORMAL,R0 ; Set success code
      5A D5 0296 728 40$: TSTL R10 ; Any pool deallocation needed?
      0D 13 0298 729 BEQL 50$ ; Br if no
      50 0F BB 029A 730 PUSHR #M<R0,R1,R2,R3> ; Save registers
      5A D0 029C 731 MOVL R10,R0 ; Get buffer address
00000000'EF 16 029F 732 JSB EXE$DEANONPAGED ; Deallocate the pool
      0F BA 02A5 733 POPR #M<R0,R1,R2,R3> ; Save registers
      05 02A7 734 50$: RSB ; Return to caller
      02AB 735 ;
      50 0C 3C 02AB 736 70$: MOVZWL #SS$_ACCVIO,R0 ; Return error code
      E9 11 02AB 737 BRB 40$ ; Return to caller
      02AD 738
```

```
02AD 740 .SBTTL SPECIAL - Handle special conditions
02AD 741
02AD 742 :++
02AD 743
02AD 744 : FUNCTIONAL DESCRIPTION:
02AD 745
02AD 746 : These routines handle data items which must be transformed
02AD 747 : before they are returned to the user. Generally, some
02AD 748 : transformation is applied to the data item and the newly
02AD 749 : computed item is stored in LOCAL_SPACE on the stack.
02AD 750 : The handling routine then changes R4 to point to LOCAL_SPACE
02AD 751 : so that MOVEIT will move the item from local storage.
02AD 752
02AD 753 : CALLING SEQUENCE:
02AD 754
02AD 755 : JSB/BSB
02AD 756
02AD 757 : INPUTS:
02AD 758
02AD 759 : R1 item identifier
02AD 760 : R3 item length
02AD 761 : R4 item address
02AD 762 : R6 user buffer length
02AD 763 : R9 LKB address
02AD 764 : R10 zero
02AD 765
02AD 766 : IMPLICIT INPUTS:
02AD 767
02AD 768 : IPL = IPL$_SYNCH
02AD 769
02AD 770 : OUTPUTS:
02AD 771
02AD 772 : R10 system buffer address to deallocate or zero if none
02AD 773
02AD 774 : IMPLICIT OUTPUTS:
02AD 775
02AD 776 : none
02AD 777
02AD 778 : ROUTINE VALUE:
02AD 779
02AD 780 : SSS_NORMAL Normal successful completion
02AD 781 : SSS_INSFMEM Insufficient non-paged dynamic memory
02AD 782
02AD 783 : SIDE EFFECTS:
02AD 784
02AD 785 : none
02AD 786 :--
02AD 787
02AD 788 : CHECK_SPC:
02AD 789
02AD 790 : : Registers R7 and R8 are saved at this level and may be used by
02AD 791 : : the action routines without being saved. Action routines are JSB'ed
02AD 792 : : to with R7 containing the address of LOCAL_SPACE on the stack.
02AD 793
02AD 794 : MOVQ R7, -(SP) ; Save registers
02AD 795 : MOVL #SPECIAL_LEN, R7 ; Get number of table entries
02AD 796 : MOVAL SPECIAL, R8 ; Get address of table
```

7E 57 7D 02AD 794 MOVQ R7, -(SP) ; Save registers
57 0A D0 02AD 795 MOVL #SPECIAL_LEN, R7 ; Get number of table entries
5B 1DA5 CF DE 02AD 796 MOVAL SPECIAL, R8 ; Get address of table

```

      88 51 B1 02B8 797 10$: CMPW R1,(R8)+      : Does entry match item?
      58 08 13 02BB 798      BEQL 20$      : Yes, go handle it
      F5 04 C0 02BD 800      ADDL #4,R8      : Skip handler address
      57 57 F5 02C0 801      SOBGTR R7,10$   : Scan rest of table
      09 11 02C3 802      BRB 30$          : Nothing to do, exit
      57 F8 AD DE 02C5 803      :
      50 01 9A 02C9 804 20$: MOVAL LOCAL_SPACE(FP),R7 : Load local address for action routine
      98 16 02CC 805      MOVZBL S^#SS$ _NORMAL,R0 : Assume success
      57 8E 7D 02CE 806      JSB @ (R8)+      : Call action routine
      05 02CE 807      :
      05 02D1 808 30$: MOVQ (SP)+,R7          : Restore registers
      02D2 809      RSB
      02D2 810      :+
      02D2 811      : Data handling routines
      02D2 812      :-
      02D2 813      :
      02D2 814      :
      02D2 815      : The PID must be returned as an EPID.
      02D2 816      : The EPID field of the LKB is valid only on a master copy lock block.
      02D2 817      :
      02D2 818      : Inputs:
      02D2 819      : R4 -> LKBSL_EPID in LKB
      02D2 820      : R7 -> Output longword buffer if needed for return
      02D2 821      : R9 = Address of LKB
      02D2 822      :
      02D2 823      : SPC_PID:
      02D2 824      : BBS #LKBSV_MSTCPY,-      : Br if master copy, R4 is pointing to
      02D4 825      : LKBSW_STATUS(R9),90$      : a valid EPID
      50 0C A9 D0 02D7 826      : MOVL LKBSL_PID(R9),R0 : Else, get the IPID
      00000000 EF 16 02DB 827      : JSB EXESL_PID_TO_EPID : Convert to EPID
      67 50 D0 02E1 828      : MOVL R0,(R7)      : Store the EPID
      54 57 D0 02E4 829      : MOVL R7,R4      : Change the item address
      50 01 9A 02E7 830 90$: MOVZBL S^#SS$ _NORMAL,R0 : Return success
      05 02EA 831      RSB
      02EB 832      :
      02EB 833      :
      02EB 834      : The lock state is a composite of several fields
      02EB 835      :
      02EB 836      :
      02EB 837      : SPC_STATE:
      02EB 838      : ASSUME LKBSB_GRMODE EQ LKBSB_RQMODE+1
      02EB 839      : ASSUME LKBSB_STATE EQ LKBSB_GRMODE+1
      67 84 3C 02EB 840      : MOVZWL (R4)+,(R7)      : Copy modes
      02 A7 64 90 02EE 841      : MOVB (R4),2(R7)      : ..and state
      05 18 02F2 842      : BGEQ 30$      : Br if state is okay
      02 A7 FF 8F 90 02F4 843      : MOVB #LKISC_WAITING,2(R7) : Else, map waiting states to same code
      54 57 D0 02F9 844 30$: MOVL R7,R4      : Change the item address
      05 02FC 845      RSB
      02FD 846      :
      02FD 847      :
      02FD 848      : The lock's parent lock ID must be extracted from another LKB
      02FD 849      :
      02FD 850      :
      02FD 851      : SPC_PARENT:
      54 67 D4 02FD 852      : CLRL (R7)      : Assume no PARENT LKB
      64 D0 02FF 853      : MOVL (R4),R4      : Get address of PARENT LKB
```



```
67 30 04 13 0302 854 BEQL 10$ : Br if none
54 57 D0 0304 855 MOVL LKBSL_LKID(R4),(R7) : Get LOCKID of owner process
D0 0308 856 10$: MOVL R7,R4 : Change the item address
05 030B 857 RSB
030C 858
030C 859 : The CSID of master
030C 860
030C 861
030C 862
030C 863 SPC_SYSTEM:
64 D5 030C 864 TSTL (R4) : Is CSID zero?
10 12 030E 865 BNEQ 30$ : Br if not, CSID is okay
50 00000000 EF D0 0310 866 MOVL L^CLUSGL_CLUB,R0 : Get address of cluster block
04 13 0317 867 BEQL 20$ : Br if no cluster
54 60 A0 9E 0319 868 MOVAB CLUBSL_LOCAL_CSID(R0),R4 : Set new item address
50 01 9A 031D 869 20$: MOVZBL S^SSS_NORMAL,R0 : Return success
05 0320 870 30$: RSB
0321 871
0321 872 : The lock's resource name space is a composite
0321 873
0321 874
0321 875
0321 876 SPC_NAMESPACE:
18 00 EF 0321 877 ASSUME RSB$B_RMOD EQ RSB$W_GROUP+2
67 64 0321 878 EXTZV #0,#8*16,- : Get the group field and access mode
64 B5 0324 879 (R4),(R7) : 3 bytes.
04 12 0326 880 TSTW (R4) : Is this group 0? (ie SYSTEM resource)
00 67 1F E2 0328 881 BNEQ 10$ : Br if not, not a system resource
54 57 D0 032A 882 BBSS #LKISV_SYSNAM,(R7),10$ : Set the SYSTEM wide indicator
05 032E 883 10$: MOVL R7,R4 : Change the item address
0331 884 RSB
0332 885
0332 886 : The lock's lock count is the sum of all locks granted on the resource.
0332 887
0332 888
0332 889
0332 890 SPC_LCKCOUNT:
58 67 D4 0332 891 CLRL (R7) : No locks granted yet!
58 54 D0 0334 892 MOVL R4,R8 : Copy listhead address
58 64 D1 0337 893 10$: CMPL (R4),R8 : Back at listhead again?
07 13 033A 894 BEQL 20$ : Br if yes
67 D6 033C 895 INCL (R7) : Else, tally one more lock
54 64 D0 033E 896 MOVL (R4),R4 : move down list
F4 11 0341 897 BRB 10$ : Look for more
54 57 D0 0343 898 20$: MOVL R7,R4 : Change item address
05 0346 899 RSB
0347 900
0347 901 : The remote lock id
0347 902
0347 903
0347 904
0347 905 SPC_REMLKID:
51 50 51 DD 0347 906 PUSHL R1 : Save R1
67 38 A9 D0 0349 907 MOVL LKBSL_RSB(R9),R1 : Get RSB address
03 A1 D0 034D 908 MOVL RSB$S_CSID(R1),(R7) : Is the REMLKID valid?
03 13 0351 909 BEQL 10$ : Br if not, information is still local
67 64 D0 0353 910 MOVL (R4),(R7) : Else, return real REMLKID
```

```
54 57 DO 0356 911 10$: MOVL R7,R4 ; Return item address
51 8E DO 0359 912 POPL R1 ; Restore R1
DS 035C 913 RSB ; Return to caller
035D 914
035D 915
035D 916
035D 917
035D 918
035D 919
06 BB 035D 920 SPC_BLOCKEDBY:
0367 30 035F 921 PUSH R1,R2 ; Save registers
2A 50 E9 0362 922 BSBW LK1_ALLOCATE ; Allocate a system buffer
58 54 DO 0365 923 BLBC R0,50$ ; Br if resource failure
54 52 DO 0368 924 MOVL R4,R8 ; Copy RSB wait queue listhead address
04 E0 036B 925 MOVL R2,R4 ; Copy address of system buffer data
12 2A A9 036D 926 BBS #LKBSV_MSTCPY,- ; Br if this is the master copy.
53 50 A9 DO 0370 927 LKBSW_STATUS(R9),10$ ; information is local to this system
53 38 A3 DO 0374 928 MOVL LKBSL_RSB(R9),R3 ; Get RSB address
08 13 0378 929 MOVL RSB$CSID(R3),R3 ; Is this a process copy?
037A 930 BEQL 10$ ; Br if not, information is still local
037A 931
037A 932 ; Lock information is on MASTER system
00000000'GF 16 037A 933 JSB G^LKISSND_BLKBY ; Send request for all locks BLOCKEDBY
03 11 0380 934 ; this lock
0380 935 BRB 30$ ; Return with status
0382 936
0382 937 ; Lock information is LOCAL to this system
0382 938
0288 30 0382 939 10$: BSBW LKISSearch_BLOCKEDBY ; Find all locks BLOCKEDBY this lock
53 53 18 B0 0385 940 30$: MOVW #LKISC_LENGTH,R3 ; Return size of item
53 53 10 78 0388 941 ASHL #16,R3,R3 ; Move to high word
53 6A B0 038C 942 MOVW (R10),R3 ; Get size of returned buffer
06 BA 038F 943 50$: POPR #^M<R1,R2> ; Restore registers
05 0391 944 RSB
0392 945
0392 946
0392 947 ; The List of all locks blocking this lock.
0392 948
0392 949
0392 950 SPC_BLOCKING:
06 BB 0392 951 PUSH R1,R2 ; Save registers
0332 30 0394 952 BSBW LK1_ALLOCATE ; Allocate a system buffer
2A 50 E9 0397 953 BLBC R0,50$ ; Br if resource failure
58 54 DO 039A 954 MOVL R4,R8 ; Copy RSB wait queue listhead address
54 52 DO 039D 955 MOVL R2,R4 ; Copy address of system buffer data
04 E0 03A0 956 BBS #LKBSV_MSTCPY,- ; Br if this is the master copy.
12 2A A9 03A2 957 LKBSW_STATUS(R9),10$ ; information is local to this system
53 50 A9 DO 03A5 958 MOVL LKBSL_RSB(R9),R3 ; Get RSB address
53 38 A3 DO 03A9 959 MOVL RSB$CSID(R3),R3 ; Is this a process copy?
08 13 03AD 960 BEQL 10$ ; Br if not, information is still local
03AF 961
03AF 962 ; Lock information is on MASTER system
03AF 963
00000000'GF 16 03AF 964 JSB G^LKISSND_BLKING ; Send request for all locks BLOCKING
03 11 03B5 965 ; this lock
03B5 966 BRB 30$ ; Return with status
03B7 967
```

```

03B7 968 ; Lock information is LOCAL to this system
03B7 969
01B2 30 03B7 970 10$: BSBW LKISSEARCH BLOCKING ; Find all locks BLOCKING this lock
53 53 18 80 03BA 971 30$: MOVW #LKISC_LENGTH,R3 ; Return size of item
53 53 10 78 03BD 972 ASHL #16,R3-R3 ; Move to high word
53 53 6A 80 03C1 973 MOVW (R10),R3 ; Get size of returned buffer
06 8A 03C4 974 50$: POPR #M<R1,R2> ; Restore registers
05 03C6 975 RSB
03C7 976
03C7 977 ; The list of all locks associated with the resource.
03C7 978
03C7 979
03C7 980
03C7 981 SPC_LOCKS:
06 8B 03C7 982 PUSHF #M<R1,R2> ; Save registers, R3 & R4 are outputs
02FD 30 03C9 983 BSBW LK1_ALLOCATE ; Allocate a system buffer
52 50 E9 03CC 984 BLBC R0,80$ ; Br if failure
58 54 D0 03CF 985 MOVL R4,R8 ; Copy listhead address
54 52 D0 03D2 986 MOVL R2,R4 ; Set address of return buffer
04 E0 03D5 987 BBS #LKBSV_MSTCPY,- ; Br if this is the master copy,
12 2A A9 03D7 988 LKBSW_STATUS(R9),10$ ; information is local to this system
53 50 A9 D0 03DA 989 MOVL LKBSL_RSB(R9),R3 ; Get RSB address
53 38 A3 D0 03DE 990 MOVL RBSL_CSID(R3),R3 ; Is this a process copy?
08 13 03E2 991 BEQL 10$ ; Br if not, information is still local
03E4 992
03E4 993 ; Lock information is on MASTER system
03E4 994
00000000'GF 16 03E4 995 JSB G^LKISND_LOCKS ; Send request for all locks associated
2B 11 03EA 996 ; with this lock
03EC 997 BRB 70$ ; Return with status
03EC 998
03EC 999 ; Lock information is LOCAL to this system
51 56 D0 03EC 1000
03EF 1001 10$: MOVL R6,R1 ; Get size of buffer
53 03 9A 03EF 1002 ASSUME RBSL_CVTQFL EQ RBSL_GRQFL+8
57 58 D0 03F2 1003 ASSUME RBSL_WTQFL EQ RBSL_CVTQFL+8
58 67 D1 03F5 1004 MOVZBL #3,R3 ; Initialize number of queues to search
14 13 03F8 1005 30$: MOVL R8,R7 ; Copy listhead address, again
51 18 C2 03FA 1006 50$: CMPL (R7),R8 ; Back at listhead again?
25 19 03FD 1007 BEQL 60$ ; Br if yes
57 67 D0 03FF 1008 SUBL #LKISC_LENGTH,R1 ; Any room left in buffer?
08 A7 9E 0402 1009 BLSS 90$ ; Br if not
23 10 0406 1010 MOVL (R7),R7 ; Else, move down list
57 38 A7 9E 0408 1011 MOVAB -LKBSL_SQFL(R7),R7 ; Point to start of LKB
E7 11 040C 1012 BSBB LOCK_INFO ; Get the lock information
040E 1013 MOVAB LKBSL_SQFL(R7),R7 ; Point back to state queue
040E 1014 BRB 50$ ; Look for more
58 08 C0 040E 1015 60$: ASSUME RBSL_CVTQFL EQ RBSL_GRQFL+8
DE 53 F5 040E 1016 ASSUME RBSL_WTQFL EQ RBSL_CVTQFL+8
50 01 9A 0411 1017 ADDL #8,R8 ; Skip to next queue
53 18 80 0414 1018 SOBGTR R3,30$ ; Loop if more queues to search
53 10 78 0417 1019 MOVZBL S^SS$_NORMAL,R0 ; Return success
53 6A 80 041A 1020 70$: MOVW #LKISC_LENGTH,R3 ; Return size of item
06 8A 041E 1021 ASHL #16,R3-R3 ; Move to high word
05 05 0421 1022 MOVW (R10),R3 ; Get size of returned buffer
0423 1023 80$: POPR #M<R1,R2> ; Restore registers
0423 1024 RSB ; Return to caller
```



```
50 0601 8F 3C 0424 1025 90$: MOVZWL #SS$_BUFFEROVF,R0 ; Return partial success
      F6 11 0424 1026 BRB 80$ ; Exit
      0429 1027
      042B 1028
      042B 1029
      042B 1030 :+ Return Lock Information
      042B 1031
      042B 1032 This routine will return the following lock information:
      042B 1033
      042B 1034 LKIS_LOCKID - the lock's lock id
      042B 1035 LKIS_PID - the lock's PID
      042B 1036 LKIS_SYSTEM - the resource's system id
      042B 1037 LKIS_STATE - the lock's current state
      042B 1038 LKIS_REMLKID - the remote lock id (Process copy LOCKID)
      042B 1039 LKIS_REMSYSTEM - the remote system id (Process copy CSID)
      042B 1040
      042B 1041 Inputs:
      042B 1042 R2 = Output buffer address
      042B 1043 R7 = LKB address
      042B 1044 R10 = Address of beginning of system buffer
      042B 1045
      042B 1046 Outputs:
      042B 1047 None
      042B 1048
      042B 1049 Side Effects:
      042B 1050 R0 is destroyed
      042B 1051 (R10) is increased by lock return size
      042B 1052
      042B 1053 LOCK_INFO:
      82 6A 18 A0 042B 1054 ADDW #LKISC_LENGTH,(R10) ; Tally return size
      82 30 A7 D0 042E 1055 MOVL LKBSL_KID(R7),(R2)+ ; Return the LOCKID (MASTER LOCKID)
      0432 1056
      0432 1057 ; The EPID in the LKB is valid only for a master lock block.
      0432 1058
      50 14 A7 D0 0432 1059 MOVL LKBSL_EPID(R7),R0 ; Get the EPID
      0A 2A A7 E0 0436 1060 BBS #LKBSV_MSTCPY,- ; Br if master copy lock
      50 0C A7 D0 0438 1061 LKBSW_STATUS(R7),10$ ; ...EPID is valid
      00000000'EF 16 043F 1062 MOVL LKBSL_PID(R7),R0 ; Get the IPID
      82 50 D0 0445 1063 JSB L^EXESIPID_TO_EPID ; Convert to EPID
      50 50 A7 D0 044B 1064 10$: MOVL R0,(R2)+ ; Return the EPID
      82 38 A0 D0 044C 1065 MOVL LKBSL_RSB(R7),R0 ; Get RSB address
      0E 12 0450 1066 MOVL RBSL_CSID(R0),(R2)+ ; Return the SYSTEM ID (MASTER CSID)
      50 00000000'EF D0 0452 1067 BNEQ 30$ ; Br if okay
      05 13 0459 1068 MOVL L^CLUSGL_CLUB,R0 ; Else, get address of cluster block
      FC A2 60 A0 D0 045B 1069 BEQL 30$ ; Br if no cluster
      82 34 A7 B0 0460 1070 MOVL CLUBSL_LOCAL_CSID(R0),-4(R2) ; Return real CSID
      82 36 A7 9B 0460 1071 30$: ASSUME LKBSB_GMODE EQ LKBSB_RMODE+1
      FE A2 FF 8F 90 0464 1072 MOVW LKBSB_RMODE(R7),(R2)+ ; Copy modes
      046B 1073 MOVZBW LKBSB_STATE(R7),(R2)+ ; Copy current state, zero byte
      046A 1074 BGEQ 40$ ; Br if state is okay
      046F 1075 MOVB #LKISC_WAITING,-2(R2) ; Else, map waiting states to same code
      046F 1076 40$:
      046F 1077 ; The remote CSID and REMLKID are only valid in a master copy
      046F 1078 ; lock block.
      82 54 A7 D0 046F 1079
      82 58 A7 D0 0473 1080 MOVL LKBSL_REMLKID(R7),(R2)+ ; Copy the REMLKID (PROCESS COPY LKID)
      0473 1081 MOVL LKBSL_CSID(R7),(R2)+ ; Get the remote CSID (PROCESS_COPY CSID)
```

SYSGETLK1
V04-000

H 3
- GET LOCK MANAGER INFORMATION SYSTEM SE 16-SEP-1984 02:18:11 VAX/VMS Macro V04-00
SPECIAL - Handle special conditions 5-SEP-1984 03:53:51 [SYS.SRC]SYSGETLK1.MAR;1

Page 22
(7)

		04	E0	0477	1082	BBS	#LKBSV_MSTCPY -	:	Br if master copy
		A7		0479	1083		LKBSW_STATUS(R7), 90\$:	...CSID, REMLKID are valid
16	2A	A7	D0	047C	1084	MOVL	LKBSL_LKID(R7), -8(R2)	:	Else, return the LOCKID as REMLKID
50	FB	A2	D0	0481	1085	MOVL	L^CLUSGL_CLUB, R0	:	Get the CLUB
	00000000	EF	13	0488	1086	BEQL	70\$:	Br if none, return zero CSID
		04	D0	048A	1087	MOVL	CLUB\$L_LOCAL_CSID(R0), R0	:	Else, get real CSID
	50	60	D0	048E	1088	MOVL	R0, -4(R2)	:	Return real CSID
	FC	A2	05	0492	1089	RSB			
		50		0493	1090				

SYSGETLK1
V04-000

```
0493 1092 .SBTTL GETLKB - Get specified Lock Block
0493 1093 ++
0493 1094
0493 1095 FUNCTIONAL DESCRIPTION:
0493 1096
0493 1097 Routine to convert a LKID and check privileges. If a valid LKID is
0493 1098 specified, the standard conversion routine VERIFYLOCKID is simply
0493 1099 called. If, however, a LKID that implies a "wildcard" LKID (-1 or 0)
0493 1100 is specified, then the next active lock is chosen as the LKID to pass
0493 1101 to VERIFYLOCKID which then checks the requestor's privilege to obtain
0493 1102 information about the lock and returns the lock's LKB address.
0493 1103
0493 1104 CALLING SEQUENCE:
0493 1105
0493 1106 JSB/BSB
0493 1107
0493 1108 INPUTS:
0493 1109
0493 1110 R4 current process PCB address
0493 1111 LKID(AP) address of specified LKID
0493 1112
0493 1113 IMPLICIT INPUTS:
0493 1114
0493 1115 IPL <= IPL$ASTDEL
0493 1116
0493 1117 OUTPUTS:
0493 1118
0493 1119 R0 success/failure of operation
0493 1120 R4 current process PCB address
0493 1121 R9 specified lock's LKB address
0493 1122
0493 1123 COMPLETION CODES:
0493 1124
0493 1125 $$$NORMAL Normal successful completion
0493 1126 $$$ACCVIO Access violation on attempt to access lock id
0493 1127 $$$NOMORELOCK No more locks available (on "wildcard" operations)
0493 1128
0493 1129 SIDE EFFECTS:
0493 1130
0493 1131 R5 and R6 are destroyed.
0493 1132 --
0493 1133
0493 1134 GETLKB:
0493 1135 CLRL R5 ; Assume not "wildcard" LKID
0493 1136 MOVL LKID(AP),R6 ; Get LKID address
0493 1137 BEQL 60$ ; Br if none
0493 1138 IFNOWRT #4,(R6),50$ ; Check access to LKID
0493 1139 MOVL (R6),R1 ; Get LKID
0493 1140 BGTR 20$ ; Br if standard LKID
0493 1141 ;
0493 1142 ; "Wildcard" type LKID specified
0493 1143 ;
0493 1144 CVTBL R1,R5 ; Get LKIX (Lock Index) from LKID
0493 1145 BGTR 10$ ; If gtr, valid LKIX
0493 1146 CLRL R5 ; Else, start with index = 1
0493 1147 INCW R5 ; Increment LKIX
0493 1148 CMPW R5,LCK$GL_MAXID ; Is LKIX in valid range?
```

56	08	AC	D0	42	13	0493	1135	CLRL	R5	:	Assume not "wildcard" LKID
						0493	1136	MOVL	LKID(AP),R6	:	Get LKID address
						0493	1137	BEQL	60\$:	Br if none
						0493	1138	IFNOWRT	#4,(R6),50\$:	Check access to LKID
51	66	D0				0493	1139	MOVL	(R6),R1	:	Get LKID
	23	14				0493	1140	BGTR	20\$:	Br if standard LKID
						0493	1141	:		:	
						0493	1142	:	"Wildcard" type LKID specified	:	
						0493	1143	:		:	
55	51	32				0493	1144	CVTBL	R1,R5	:	Get LKIX (Lock Index) from LKID
	02	14				0493	1145	BGTR	10\$:	If gtr, valid LKIX
	55	D4				0493	1146	CLRL	R5	:	Else, start with index = 1
	55	B6				0493	1147	INCW	R5	:	Increment LKIX
00000000'EF	55	B1				0493	1148	CMPW	R5,LCK\$GL_MAXID	:	Is LKIX in valid range?

50		00000000	'FF	45	1A	04B6	1149	BGTRU	60\$:	Br if not - no more locks
				EB	D0	04B8	1150	MOVL	@LKB\$GL_IDTBL[R5],R0	:	Get LKB address
					18	04C0	1151	BGEQ	10\$:	Br if unused slot
		51	30	A0	D0	04C2	1152	MOVL	LKB\$L_LKID(R0),R1	:	Get LKID from LKB
		66		51	D0	04C6	1153	MOVL	R1,(R6)	:	Store LKID in argument list
						04C9	1154	:		:	
						04C9	1155	:	Get LKB and check privileges	:	
						04C9	1156	:		:	
			0018		30	04C9	1157	BSBW	VERIFYLOCKID	:	Get LKB address and check privileges
			55		B5	04CC	1158	TSTM	R5	:	"wildcard" type LKID specified?
			07		13	04CE	1159	BEQL	40\$:	Br if not
			DA	50	E9	04D0	1160	BLBC	R0,10\$:	Br if error, return only "good" ones
		02	A6	01	AE	04D3	1161	MNEGW	#1,2(R6)	:	Else, set continuation context
					05	04D7	1162	RSB		:	Return to caller
						04D8	1163			:	
			50		3C	04D8	1164	MOVZWL	#SS\$_ACCVIO,R0	:	Set access violation
					11	04DB	1165	BRB	40\$:	
										:	
			50		3C	04DD	1166	MOVZWL	#SS\$_NOMORELOCK,R0	:	Set no more processes
					11	04E2	1167	BRB	40\$:	
						04E4	1168			:	

[illegible]

PSEC

SAB
WSY
YEX

```
04E4 1170      .SBTTL VERIFYLOCKID - Verify lock id
04E4 1171
04E4 1172      :++
04E4 1173      FUNCTIONAL DESCRIPTION:
04E4 1174      :
04E4 1175      This routine verifies a lock id for correct process ownership
04E4 1176      and access mode and then converts it into a LKB address.
04E4 1177      :
04E4 1178      LKB is not locked after leaving this routine, therefore we
04E4 1179      must re-verify the LKB everytime we attempt to use it.
04E4 1180
04E4 1181      CALLING SEQUENCE:
04E4 1182      :
04E4 1183      JSB/BSB
04E4 1184      :
04E4 1185      Note: IPL is raised to IPL$ SYNCH to prevent the owner of
04E4 1186      the lock from releasing the LKB/RSB in the middle of verifying
04E4 1187      its lock id.
04E4 1188
04E4 1189      INPUTS:
04E4 1190      :
04E4 1191      R1      Lock id
04E4 1192      R4      Address of PCB
04E4 1193      R5      Zero if not a wildcard search operation
04E4 1194
04E4 1195      OUTPUTS:
04E4 1196      :
04E4 1197      R0      Completion code
04E4 1198      R9      Address of LKB
04E4 1199
04E4 1200      COMPLETION CODES:
04E4 1201      :
04E4 1202      SSS_NORMAL      Lock id was valid and converted to LKB address
04E4 1203      SSS_IVLOCKID    Invalid lock id
04E4 1204      SSS_IVMODE      Access mode violation on attempt to access lock
04E4 1205      SSS_NOSYSLCK    No SYSLCK privilege to access system lock
04E4 1206      SSS_NOWORLD    No WORLD privilege to access lock
04E4 1207
04E4 1208      SIDE EFFECTS:
04E4 1209      :
04E4 1210      R0 and R1 are destroyed
04E4 1211      :--
04E4 1212
04E4 1213      ASSUME LKBSV_MODE EQ 0
04E4 1214      ASSUME LKBSS_MODE EQ 2
04E4 1215
04E4 1216      VERIFYLOCKID:
04E4 1217      DSBINT #IPL$ SYNCH      : Raise IPL to sync access to LKBs
04E4 1218      MOVZWL R1,R9            : Put lockid index in R9
04E4 1219      CMPL   R9,LCK$GL_MAXID  : Is the lock id too big?
04E4 1220      BGTRU   40$            : Yes
04E4 1221      MOVL   @LCK$GL_IDTBL[R9],R9 : Get LKB address
04E4 1222      BGEQ   40$            : Unallocated id
04E4 1223      CMPL   R1,LKBSL_LKID(R9) : Check sequence number
04E4 1224      BNEQ   40$            : Not valid
04E4 1225      MOVL   LKBSL_RSB(R9),R0  : Get RSB address
04E4 1226      TSTW   RBSW_GROUP(R0)   : Is this a system resource?
```

```
59 51 3C
00000000'EF 59 D1
5A 1A
59 00000000'FF 49 D0
50 18
30 A9 51 D1
4A 12
50 50 A9 D0
4C A0 B5
050A 1226
```

```
51 00000000'GF 17 13 050D 1227 BEQL 10$ : Br if yes
      00BE C1 B1 050F 1228 MOVL G^SCH$GL CURPCB,R1 : Else, get our PCB address
      4C A0 13 0516 1229 CMPW PCB$W_GRP(R1),- : Do we have group access to LKB?
      1A 13 051A 1230 RSB$W_GROUP(R0) : ..no privilege needed
      12 11 051C 1231 BEQL 20$ : Br if our group - always allowed
      50 DC 051E 1232 IFNPRIV WORLD,70$ : Br if NO privilege to access lock
      16 EF 0524 1233 BRB 20$ : Else, success
      50 50 02 0526 1234 10$: MOVPSL R0 : Get current PSL
      05 05 02 0528 1235 EXTZV #PSL$V_PVMOD,- : Extract previous mode field
      50 50 02 052A 1236 #PSL$S_PVMOD,R0,R0
      05 01 91 052D 1237 ASSUME PSL$C_KERNEL EQ 0
      06 1E 052D 1238 ASSUME PSL$C_EXEC EQ 1
      50 06 1E 052D 1239 CMPB #PSL$C_EXEC,R0 : Does the user have the right access
      50 DC 0530 1240 BGEQU 20$ : mode to access the LKB?
      50 16 EF 0530 1241 IFNPRIV SYSLCK,60$ : Br if yes
      50 50 02 0532 1242 MOVPSL R0 : Br if NO privilege to look at lock
      51 50 50 A9 D0 0538 1243 20$: EXTZV #PSL$V_PVMOD,- : Get current PSL
      4E A1 50 053C 1244 #PSL$S_PVMOD,R0,R0 : Extract previous mode field
      50 01 9A 053F 1245 MOVL LKB$L_RSB(R9),R1 : Get RSB address
      50 0E 1A 0543 1246 CMPB R0,RSB$B_RMOD(R1) : Caller have privilege to access lock?
      50 01 9A 0547 1247 BGTRU 50$ : Br if No
      05 05 05 0549 1248 MOVZBL S^#SS$_NORMAL,R0 : Else, Yes - return success
      05 05 05 054C 1249 ENBINT : Restore IPL
      05 05 05 054F 1250 RSB
      50 2124 8F 3C 0550 1251 30$: MOVZWL #SS$_IVLOCKID,R0
      50 0354 8F 3C 0550 1252 40$: BRB 30$ : Invalid lock id
      50 28F4 8F 3C 0555 1253 50$: MOVZWL #SS$_IVMODE,R0 : Leave
      50 2884 8F 3C 0557 1254 60$: BRB 30$ : Illegal access mode
      50 2884 8F 3C 055C 1255 70$: BRB 30$ : Leave
      50 2884 8F 3C 055E 1256 60$: MOVZWL #SS$_NOSYSLCK,R0 : No SYSLCK privilege to access lock
      50 2884 8F 3C 0563 1257 70$: BRB 30$ : Leave
      50 2884 8F 3C 0565 1258 70$: MOVZWL #SS$_NOWORLD,R0 : No WORLD privilege to access lock
      50 2884 8F 3C 0565 1259 70$: BRB 30$ : Leave
      50 2884 8F 3C 056A 1260
      50 2884 8F 3C 056C 1261
```



```
056C 1263 .SBTTL LKISSEARCH_BLOCKING - Search for locks blocking the current lock
056C 1264
056C 1265 :++
056C 1266 : FUNCTIONAL DESCRIPTION:
056C 1267 :
056C 1268 : This routine searches for locks blocking the current lock. A
056C 1269 : blocking lock is one in which the maximized request mode is
056C 1270 : incompatible with the requested mode (if the lock is on the
056C 1271 : waiting or conversion queue) or the granted mode (if the lock
056C 1272 : is on the granted queue).
056C 1273 :
056C 1274 : For example, assume there is PR locks granted on a resource and
056C 1275 : a second user issues an EX mode request on the resource. The first
056C 1276 : lock is now BLOCKING the second lock and the first lock would be
056C 1277 : returned in list of locks BLOCKING the second lock.
056C 1278 :
056C 1279 : To find BLOCKING locks it is sufficient to check all locks
056C 1280 : ahead of this lock on all queues (in th order, REQUESTED,
056C 1281 : CONVERSION and then GRANTED) to see if their requested or granted
056C 1282 : modes are incompatible with this locks requested mode.
056C 1283 :
056C 1284 : CALLING SEQUENCE:
056C 1285 :
056C 1286 : JSB/BSB
056C 1287 :
056C 1288 : INPUTS:
056C 1289 :
056C 1290 : R2 address of system buffer for storing the lock information
056C 1291 : R6 length of system buffer for storing the lock information
056C 1292 : R8 address of wait queue in RSB
056C 1293 : R9 LKB address
056C 1294 :
056C 1295 : IMPLICIT INPUTS:
056C 1296 :
056C 1297 : IPL = IPL$_SYNCH
056C 1298 :
056C 1299 : OUTPUTS:
056C 1300 :
056C 1301 : R0 always success!
056C 1302 :
056C 1303 : SIDE EFFECTS:
056C 1304 :
056C 1305 : R7 is destroyed.
056C 1306 : --
056C 1307 :
056C 1308 LKISSEARCH_BLOCKING::
0066 8F BB 056C 1309 PUSHF #M<R1,R2,R5,R6> ; Save registers
0570 1310 :
0570 1311 : First run through all locks waiting ahead of this lock
0570 1312 : maximizing the requested modes and checking all locks
0570 1313 : incompatible with the current "maxmode". If this lock is
0570 1314 : on the wait queue then we do the wait queue first and
0570 1315 : the conversion queue next. If this lock is on the
0570 1316 : conversion queue then we do only the conversion queue.
0570 1317 : Later we'll do all the granted locks.
0570 1318 :
0570 1319 : If this lock is on the granted queue, we skip right to the
```

```
0570 1320      : search of the granted queue locks.
0570 1321      :
0570 1322      ASSUME LKBSK_GRANTED EQ 1
0570 1323      ASSUME LKBSK_CONVERT EQ 0
0570 1324      ASSUME LKBSK_WAITING EQ -1
0570 1325      ASSUME RSB$$_CVTQFL EQ RSB$$_GRQFL+8
0570 1326      ASSUME RSB$$_WTQFL EQ RSB$$_CVTQFL+8
0570 1327      :
55 34 A9 9A 0570 1328      MOVZBL LKBSB_RQMODE(R9),R5      : Get the current lock's requested mode
57 59 D0 0574 1329      MOVL R9,R7      : R7 will point to other LKB's
      : in front of the one pointed to by R9
      36 A9 95 0577 1331      TSTB LKBSB_STATE(R9)      : Which queue is lock on?
      63 14 057A 1332      BGTR 60$      : Br if granted queue
      03 19 057C 1333      BLSS 10$      : Br if waiting queue
      057E 1334      :
      057E 1335      : Lock is on the conversion queue
      057E 1336      :
58 08 C2 057E 1337      SUBL #8,R8      : Point to conversion queue header
      0581 1338      :
57 3C A7 D0 0581 1339 10$:      MOVL LKBSL_SQBL(R7),R7      : Get previous lock on state queue
58 57 D1 0585 1340      CMPL R7,R8      : Reached head of queue yet?
      42 13 0588 1341      BEQL 50$      : Br if yes
      57 38 C2 058A 1342      SUBL #LKBSL_SQFL,R7      : Back up to point at start of LKB
50 34 A7 9A 058D 1343      MOVZBL LKBSB_RQMODE(R7),R0      : R0 = requested mode
51 55 D0 0591 1344      MOVL R5,R1      : Save old maxmode
      0594 1345      :
      0594 1346      : Maximize lock modes (in R0 and R5) and see if this lock (R7) is
      0594 1347      : incompatible with (the previous) maxmode. The maximization function
      0594 1348      : is a simple arithmetic maximum except if the two modes are CW and PR.
      0594 1349      : In that case the maximum of CW and PR is PW. PW is incompatible
      0594 1350      : with everything either CW or PR is incompatible with.
      0594 1351      :
55 50 91 0594 1352      CMPB R0,R5      : Current mode greater than maxmode?
      20 13 0597 1353      BEQL 35$      : Br if No, they're equal
      0C 1A 0599 1354      BGTRU 20$      : Br if Yes, compute new maxmode
02 50 91 0598 1355      CMPB R0,#LKBSK_CWMODE      : Br if No, is current mode CW?
      19 12 059E 1356      BNEQ 35$      : Br if No, maxmode = R2
03 55 91 05A0 1357      CMPB R5,#LKBSK_PMODE      : Br if Yes, is maxmode PR?
      14 12 05A3 1358      BNEQ 35$      : Br if No, maxmode = R2
      0A 11 05A5 1359      BRB 25$      : Br if Yes, new maxmode is PW
02 55 91 05A7 1360 20$:      CMPB R5,#LKBSK_CWMODE      : Is maxmode CW?
      0A 12 05AA 1361      BNEQ 30$      : Br if No, maxmode = R0
03 50 91 05AC 1362      CMPB R0,#LKBSK_PMODE      : Br if Yes, is current mode PR?
      05 12 05AF 1363      BNEQ 30$      : Br if No, maxmode = R0
55 04 90 05B1 1364 25$:      MOVB #LKBSK_PMODE,R5      : Have CW and PR; maxmode = PW
      03 11 05B4 1365      BRB 35$      :
55 50 90 05B6 1366 30$:      MOVB R0,R5      : Maxmode = R0
      05B9 1367      :
00000000'EF41 50 E0 05B9 1368 35$:      BBS R0,-      : Branch if compatible with
      BF      L*LCK$COMPAT_TBL[R1],10$; saved maxmode
      05C1 1369      :
      05C2 1370      : Have a lock incompatible with maxmode, return the lock info.
      05C2 1371      :
      05C2 1372      :
56 18 C2 05C2 1373      SUBL #LKISC_LENGTH,R6      : Any room left in buffer?
      3E 19 05C5 1374      BLSS 90$      : Br if not, leave now
      FE61 30 05C7 1375      BSBW LOCK_INFO      : Return the lock information
      B5 11 05CA 1376 40$:      BRB 10$      : Get next lock in RSB (outer loop)
```

```
05CC 1377
05CC 1378 50$:
05CC 1379
05CC 1380
05CC 1381
05CC 1382
05CC 1383
05CC 1384
05CC 1385
57 58 08 C2 05CC 1386
   C8 A8 9E 05CF 1387
   10 C1 05D3 1388
50 50 A9 05D5 1389
   50 58 D1 05D8 1390
   ED 12 05DB 1391
05DD 1392
05DD 1393
05DD 1394
05DD 1395
03 11 05DD 1396
05DF 1397
05DF 1398 60$:
05DF 1399
05DF 1400
58 10 C2 05DF 1401
05E2 1402
57 3C A7 D0 05E2 1403 70$:
58 57 D1 05E6 1404
   1A 13 05E9 1405
57 38 C2 05EB 1406
50 35 A7 9A 05EE 1407
E7 00000000'EF45 50 E0 05F2 1408
05FB 1409
05FB 1410
05FB 1411
56 18 C2 05FB 1412
   05 19 05FE 1413
FE28 30 0600 1414
   DD 11 0603 1415
0605 1416
50 01 9A 0605 1417 90$:
0066 BF BA 0608 1418
   05 060C 1419
060D 1420

: Reached the queue header. Back up R8 to point to the previous
: queue header in the RSB. If R8 is pointing to the granted
: queue, then we are done with this loop and we continue with
: the granted queue. Otherwise, we repeat this loop for the
: conversion queue.
SUBL #8,R8 ; Back up R8 one queue header
MOVAB -LKBSL_SQFL(R8),R7 ; Prepare to process that queue
ADDL3 #RSBSL_GRQFL,- ; Get address of granted queue
      LKBSL_RSB(R9),R0
CMPL R8,R0 ; Have we reached the granted queue?
BNEQ 40$ ; Br if Not, repeat for conversion queue

: Now repeat a similar procedure for all locks on the granted
: queue whose granted mode is incompatible with the maxmode
: in R5.
BRB 70$

: Lock is initially on the granted queue.
SUBL #16,R8 ; Point to granted queue header

MOVL LKBSL_SQBL(R7),R7 ; Get next lock in granted queue
CMPL R7,R8 ; Reached end of queue?
BEQL 90$ ; Br if Yes, all done
SUBL #LKBSL_SQFL,R7 ; Back up to point at start of LKB
MOVZBL LKBSL_GMODE(R7),R0 ; Get granted mode
BBS R0,L^CCK$COMPAT_TBL[R5],70$ ; Branch if compatible

: Have an incompatible lock on the granted queue, return lock info.
SUBL #LKISC_LENGTH,R6 ; Any room left in buffer?
BLSS 90$ ; Br if not, leave now
BSBW LOCK_INFO ; Return lock info
BRB 70$ ; Look for more

MOVZBL #1,R0 ; Success indicator
POPR #^M<R1,R2,R5,R6> ; Restore registers
RSB
```



```
060D 1422 .SBTTL LKIS$SEARCH_BLOCKEDBY - Search for locks blockedby the current lock
060D 1423
060D 1424 :++
060D 1425 : FUNCTIONAL DESCRIPTION:
060D 1426 :
060D 1427 : This routine searches for locks blocked by the current lock.
060D 1428 : A blocked lock is one which is either blocked by the current
060D 1429 : lock or is blocked by any other lock blocked by the current
060D 1430 : lock. We must start with the current lock on whatever queue
060D 1431 : it may currently be on and then maximize the requested for
060D 1432 : locks on the converting or waiting queues. All locks are checked
060D 1433 : to see if the maximized request mode is incompatible with the
060D 1434 : requested mode (if the locks is not on the granted queue).
060D 1435 :
060D 1436 : For example, assume there is an EX lock granted on a resource and
060D 1437 : a two other users have issued PR requests on the resource. Now
060D 1438 : if we wish to find all locks BLOCKEDBY the first lock, then the
060D 1439 : list consists of the two locks waiting for the resource in PR
060D 1440 : mode.
060D 1441 :
060D 1442 : To find BLOCKING locks it is sufficient to check all locks
060D 1443 : behind the current lock on all queues (in the order, GRANTED
060D 1444 : CONVERTING and then WAITING) to see if their requested mode
060D 1445 : is incompatible with the current lock's requested (or granted)
060D 1446 : mode. Once, we have found one blocked lock, then that lock and all
060D 1447 : locks following are also blocked.
060D 1448 :
060D 1449 : CALLING SEQUENCE:
060D 1450 :
060D 1451 : JSB/BSB
060D 1452 :
060D 1453 : INPUTS:
060D 1454 :
060D 1455 : R2 address of system buffer for storing the lock information
060D 1456 : R6 length of system buffer for storing the lock information
060D 1457 : R8 address of wait queue in RSB
060D 1458 : R9 LKB address
060D 1459 :
060D 1460 : IMPLICIT INPUTS:
060D 1461 :
060D 1462 : IPL = IPL$_SYNCH
060D 1463 :
060D 1464 : OUTPUTS:
060D 1465 :
060D 1466 : R0 always success!
060D 1467 :
060D 1468 : SIDE EFFECTS:
060D 1469 :
060D 1470 : R7 is destroyed.
060D 1471 : --
060D 1472 :
060D 1473 LKIS$SEARCH_BLOCKEDBY::
0066 8F BB 060D 1474 PUSHF #^M<R1,R2,R5,R6> ; Save registers
0611 1475 :
0611 1476 : First run through all locks waiting behind this lock
0611 1477 : maximizing the requested modes and checking all locks
0611 1478 : incompatible with the current 'maxmode'. If we find a
```

```
0611 1479 ; Lock that is blocked by the current lock, then that lock
0611 1480 ; and all the following locks are blocked. For locks that
0611 1481 ; are on the granted queue we do not maximize the granted
0611 1482 ; mode, for all other queues we will maximize the request mode.
0611 1483 ;
0611 1484 ; If this lock is not on the granted queue, we skip right to the
0611 1485 ; search of the other queue locks.
0611 1486 ;
0611 1487 ASSUME LKBSK_GRANTED EQ 1
0611 1488 ASSUME LKBSK_CONVERT EQ 0
0611 1489 ASSUME LKBSK_WAITING EQ -1
0611 1490 ASSUME RBSL_CVTQFL EQ RBSL_GRQFL+8
0611 1491 ASSUME RBSL_WTQFL EQ RBSL_CVTQFL+8
0611 1492
55 34 A9 9A 0611 1493 MOVZBL LKBSB_RQMODE(R9),R5 ; Get the current lock's requested mode
57 59 D0 0615 1494 MOVL R9,R7 ; R7 will point to other LKB's
0618 1495 ; after the one pointed to by R9
36 A9 95 0618 1496 TSTB LKBSB_STATE(R9) ; Which queue is lock on?
21 19 061B 1497 BLSS 20$ ; Br if waiting
22 13 061D 1498 BEQL 30$ ; Br if converting
061F 1499 ;
061F 1500 ; Lock is on the granted queue
061F 1501 ;
55 35 A9 9A 061F 1502 MOVZBL LKBSB_GRMODE(R9),R5 ; Get the current lock's granted mode
0623 1503 ;
57 38 A7 D0 0623 1504 10$: MOVL LKBSL_SQFL(R7),R7 ; Get next lock on state queue
58 57 D1 0627 1505 CMPL R7,R8 ; Reached head of queue yet?
58 13 062A 1506 BEQL 90$ ; Br if yes
57 38 C2 062C 1507 SUBL #LKBSL_SQFL,R7 ; Back up to point at start of LKB
50 35 A7 9A 062F 1508 MOVZBL LKBSB_GRMODE(R7),R0 ; Get the lock's granted mode
00000000'EF45 50 E0 0633 1509 BBS R0,- ; Branch if compatible
E7 063B 1510 L^LK$COMPAT_TBL[R5],10$;
063C 1511 ;
063C 1512 ; Have an incompatible, return the lock info. for all succeeding locks
063C 1513 ;
62 11 063C 1514 BRB 120$ ; Return lock info.
063E 1515 ;
063E 1516 20$: ; Lock is initially on the waiting queue.
063E 1517 ;
58 08 C0 063E 1519 ADDL #8,R8 ; Advance R8 one queue header
0641 1520 30$: ;
0641 1521 ; Lock is initially on the converting queue, OR we have
0641 1522 ; reached the queue header. Advance R8 to point to the next
0641 1523 ; queue header in the RSB.
0641 1524 ;
58 08 C0 0641 1525 ADDL #8,R8 ; Advance R8 one queue header
0644 1526 ;
0644 1527 ; Run thru all locks on either the converting or waiting queue
0644 1528 ; lock for any locks blocked by the maxmode in R5.
0644 1529 ;
57 38 A7 D0 0644 1530 40$: MOVL LKBSL_SQFL(R7),R7 ; Get next lock in queue
58 57 D1 0648 1531 CMPL R7,R8 ; Reached end of queue?
58 3A 13 064B 1532 BEQL 90$ ; Br if Yes, all done
57 38 C2 064D 1533 SUBL #LKBSL_SQFL,R7 ; Back up to point at start of LKB
50 34 A7 9A 0650 1534 MOVZBL LKBSB_RQMODE(R7),R0 ; Get requested mode
51 55 D0 0654 1535 MOVL R5,R1 ; Save old maxmode
```

```
0657 1536
0657 1537
0657 1538
0657 1539
0657 1540
0657 1541
0657 1542
55 50 91 0657 1543
20 13 065A 1544
0C 1A 065C 1545
02 50 91 065E 1546
19 12 0661 1547
03 55 91 0663 1548
14 12 0666 1549
0A 11 0668 1550
02 55 91 066A 1551 50$:
0A 12 066D 1552
03 50 91 066F 1553
05 12 0672 1554
55 04 90 0674 1555 60$:
03 11 0677 1556
55 50 90 0679 1557 70$:
067C 1558
00000000'EF41 50 E1 067C 1559 80$:
1B 0684 1560
BD 11 0685 1561
0687 1562
58 08 C0 0687 1563 90$:
57 C8 A8 9E 068A 1564
28 C1 068E 1565
50 50 A9 0690 1566
50 58 D1 0693 1567
AC 12 0696 1568
0698 1569
50 01 9A 0698 1570 100$:
0066 8F BA 069B 1571
05 069F 1572
06A0 1573
06A0 1574
06A0 1575
06A0 1576
56 18 C2 06A0 1577 120$:
F3 19 06A3 1578
FD83 30 06A5 1579
57 38 A7 D0 06A8 1580 130$:
58 57 D1 06AC 1581
05 13 06AF 1582
57 38 C2 06B1 1583
EA 11 06B4 1584
06B6 1585
58 08 C0 06B6 1586 140$:
57 C8 A8 9E 06B9 1587
28 C1 06BD 1588
50 50 A9 06BF 1589
50 58 D1 06C2 1590
D1 13 06C5 1591
DF 11 06C7 1592

: Maximize lock modes (in R0 and R5) and see if this lock (R7) is
: incompatible with (the previous) maxmode. The maximization function
: is a simple arithmetic maximum except if the two modes are CW and PR.
: In that case the maximum of CW and PR is PW. PW is incompatible
: with everything either CW or PR is incompatible with.
CMPB R0,R5 : Current mode greater than maxmode?
BEQL 80$ : Br if No, they're equal
BGTRU 50$ : Br if Yes, compute new maxmode
CMPB R0,#LCK$K_CWMODE : Br if No, is current mode CW?
BNEQ 80$ : Br if No, maxmode = R2
CMPB R5,#LCK$K_PMODE : Br if Yes, is maxmode PR?
BNEQ 80$ : Br if No, maxmode = R2
BRB 60$ : Br if Yes, new maxmode is PW
CMPB R5,#LCK$K_CWMODE : Is maxmode CW?
BNEQ 70$ : Br if No, maxmode = R0
CMPB R0,#LCK$K_PMODE : Br if Yes, is current mode PR?
BNEQ 70$ : Br if No, maxmode = R0
MOVB #LCK$K_PMODE,R5 : Have CW and PR; maxmode = PW
BRB 80$
MOVVB R0,R5 : Maxmode = R0
BBC R0,- : Branch if incompatible
L^LCK$COMPAT_TBL[R1],120$ : with saved maxmode
BRB 40$ : Else, check next lock in RSB
ADDL #8,R8 : Advance R8 one queue header
MOVAB -LKB$SL_SQFL(R8),R7 : Prepare to process that queue
ADDL3 #RSB$SL_WTQFL+8,- : Get address past waiting queue
LKB$SL_RSB(R9),R0
CMPL R8,R0 : Have we done all the queues?
BNEQ 40$ : Br if Not, repeat for remaining queue
MOVZBL #1,R0 : Success indicator
POPR #^M<R1,R2,R5,R6> : Restore registers
RSB
: We have found the first incompatible lock
SUBL #LKISC_LENGTH,R6 : Any room left in buffer?
BLSS 100$ : Br if not
BSBW LOCK_INFO : Else, return lock info.
MOVL LKB$SL_SQFL(R7),R7 : Get next lock in queue
CMPL R7,R8 : Reached end of queue?
BEQL 140$ : Br if Yes, skip to next queue
SUBL #LKB$SL_SQFL,R7 : Back up to point at start of LKB
BRB 120$ : Return the lock info.
ADDL #8,R8 : Advance R8 one queue header
MOVAB -LKB$SL_SQFL(R8),R7 : Prepare to process that queue
ADDL3 #RSB$SL_WTQFL+8,- : Get address past end of queues
LKB$SL_RSB(R9),R0
CMPL R8,R0 : Have we done all queues?
BEQL 100$ : Br if Yes, leave
BRB 130$ : Else, loop thru remaining queues
```


SYSGETLKI
V04-000

- GET LOCK MANAGER INFORMATION SYSTEM SE 16-SEP-1984 02:18:11 VAX/VMS Macro V04-00
LKISSEARCH_BLOCKEDBY - Search for locks 5-SEP-1984 03:53:51 [SYS.SRC]SYSGETLKI.MAR;1

Page 33
(11)

0609 1593

SYS
V04-

```
06C9 1595      .SBTTL LKI_ALLOCATE - Allocate a system buffer
06C9 1596
06C9 1597      ++
06C9 1598      FUNCTIONAL DESCRIPTION:
06C9 1599      This routine attempts to allocate a system buffer and initialize
06C9 1600      the structure type.
06C9 1601
06C9 1602      CALLING SEQUENCE:
06C9 1603      JSB/BSB
06C9 1604
06C9 1605      INPUTS:
06C9 1606      R6      Size of desired buffer minus header
06C9 1607
06C9 1608      IMPLICIT INPUTS:
06C9 1609      IPL = IPL$_SYNCH
06C9 1610
06C9 1611      OUTPUTS:
06C9 1612      R0      Completion status for request
06C9 1613      R2      Address of the system buffer at data portion of buffer
06C9 1614      R10     Address of start of the system buffer
06C9 1615
06C9 1616      SIDE EFFECTS:
06C9 1617      none
06C9 1618      --
06C9 1619
06C9 1620      LKI_ALLOCATE:
06C9 1621      PUSH  R1,R3,R4      ; Save registers
06C9 1622      MOVL  SCH$GL_CURPCB,R4 ; Get PCB address
06C9 1623      ADDL  #12,R6,R1      ; Compute size of system buffer
06C9 1624
06C9 1625      ; NOTE: The exec routine EXES$BUFFRQUOTA cannot be called, since
06C9 1626      ; it will lower IPL and destroy all synchronization.
06C9 1627
06C9 1628      MOVZWL IOCS$GW_MAXBUF,R0 ; Get maximum buffer size allowed
06C9 1629      CMPL  R1,R0          ; Is buffer too big?
06C9 1630      BGTRU 20$            ; Br if yes, error
06C9 1631      MOVL  PCB$JIB(R4),R0 ; Get JIB address
06C9 1632      CMPL  R1,JIB$$_BYTLM(R0) ; Is BYTLM quota okay?
06C9 1633      BGTRU 20$            ; Br if not, error
06C9 1634      CMPL  R1,JIB$$_BYTCNT(R0) ; Is BYTCNT quota okay?
06C9 1635      BGTRU 20$            ; Br if not, error
06C9 1636      JSB   EXES$ALONONPAGED ; Try and allocate a buffer
06C9 1637      BLBC  R0,30$         ; Br if failed
06C9 1638      MOVL  R2,R10        ; Set address of buffer to deallocate
06C9 1639
06C9 1640      ; Initialize structure header
06C9 1641
06C9 1642      CLRQ  (R2)+          ; Zero return size, unused fields
06C9 1643      MOVW  R1,(R2)+       ; Set structure size
06C9 1644      MOVW  #DYN$C_BUF10,(R2)+ ; Set structure type
06C9 1645      POPR  #R1,R3,R4      ; Restore registers
06C9 1646
06C9 1647      10$:
```

54	00000000	1A	BB	06C9	1627				
	51	56	OC	06CB	1628				
				06D2	1629				
				06D6	1630				
				06D6	1631				
				06D6	1632				
				06D6	1633				
50	00000000	EF	3C	06D6	1634				
	50	51	D1	06DD	1635				
		28	1A	06E0	1636				
50	0080	C4	D0	06E2	1637				
	24	A0	51	06E7	1638				
		1D	1A	06EB	1639				
	20	A0	51	06ED	1640				
		17	1A	06F1	1641				
	00000000	EF	16	06F3	1642				
		13	50	06F9	1643				
	5A	52	D0	06FC	1644				
				06FF	1645				
				06FF	1646				
				06FF	1647				
	82	82	7C	06FF	1648				
	82	51	B0	0701	1649				
	82	13	B0	0704	1650				
		1A	BA	0707	1651	10\$:			

SYSGETLKI
V04-000

H 4
- GET LOCK MANAGER INFORMATION SYSTEM SE 16-SEP-1984 02:18:11 VAX/VMS Macro V04-00
LKI_ALLOCATE - Allocate a system buffer 5-SEP-1984 03:53:51 [SYS.SRC]SYSGETLKI.MAR;1

Page 35
(12)

```

      05 0709 1652      RSB
      070A 1653
50      1C 3C 070A 1654 20$: MOVZWL #SS$_EXQUOTA,R0      ; Set error return
      FB 11 070D 1655      BRB 10$      ; Return to caller
      070F 1656
50      0124 8F 3C 070F 1657 30$: MOVZWL #SS$_INSFMEM,R0      ; Set error return
      F1 11 0714 1658      BRB 10$      ; Return to caller
      0716 1659
      0716 1660
      0716 1661      .END
```


SYSGETLKI
Symbol table

- GET LOCK MANAGER INFORMATION SYSTEM SE 16-SEP-1984 02:18:11 VAX/VMS Macro V04-00
5-SEP-1984 03:53:51 [SYS.SRC]SYSGETLKI.MAR;1

Page 36
(12)

```

$ST1 = 00000000
ACBSL_KAST = 00000018
ACB_L_COUNT = 0000002C
ACB_L_DADDR = 0000001C
ACB_L_EFN = 00000020
ACB_L_ILIST = 00000030
ACB_L_IOSB = 00000024
ACB_L_OPID = 00000028
ASTADR = 00000014
ASTPRM = 00000018
BSTRING = 00000001
CHECKITEM = 000001F3 R 02
CHECK_SPC = 000002AD R 02
CLUSGC_CLUB = ***** X 02
CLUBSL_LOCAL_CSID = 00000060
CSTRING = 00000002
DYN$C_BUFIO = 00000013
EFN = 00000004
EXESALONONPAGED = ***** X 02
EXESDEANONPAGED = ***** X 02
EXESGETLKI = 00000000 RG 03
EXESIPID_TO_EPID = ***** X 02
EXESPROBEW = ***** X 02
EXE_GETLKI = 00000098 R 02
GET[KB = 00000493 R 02
GET_REMLKI = 000001D0 R 02
GRET = 00000177 R 02
IOCSGW_MAXBUF = ***** X 02
IOSB = 00000010
IPL$ASTDEL = 00000002
IPL$SYNCH = 00000008
ITMLST = 0000000C
JIBSL_BYTCNT = 00000020
JIBSL_BYTLM = 00000024
LCK$CHECK_STALL = ***** X 02
LCK$COMPAT_TBL = ***** X 02
LCK$GL_IDTBL = ***** X 02
LCK$GL_MAXID = ***** X 02
LCK$K_CWMODE = 00000002
LCK$K_PMODE = 00000003
LCK$K_PWMODE = 00000004
LIMSG$K_ZERO = 00000000
LIMSG$SL_LCKCOUNT = 0000002C
LIMSG$SL_RSBREFCNT = 00000028
LIMSG$SL_STATE = 00000024
LIMSG$Q_VALBLK = 00000030
LKBSB_GRMODE = 00000035
LKBSB_RQMODE = 00000034
LKBSB_STATE = 00000036
LKBSK_CONVERT = 00000000
LKBSK_GRANTED = 00000001
LKBSK_WAITING = 00000000
LKBSL_CSID = 00000058
LKBSL_EPID = 00000014
LKBSL_LKID = 00000030
LKBSL_PARENT = 00000048
LKBSL_PID = 0000000C

```

```

LKBSL_REMLKID = 00000054
LKBSL_RSB = 00000050
LKBSL_SQBL = 0000003C
LKBSL_SQFL = 00000038
LKBS$MODE = 00000002
LKBSV_MODE = 00000000
LKBSV_MSTCPY = 00000004
LKBSW_REFcnt = 0000004C
LKBSW_STATUS = 0000002A
LKBTBC = 00000002 R 02
LKISC_LENGTH = 00000018
LKISC_LKBTYP = 00000001
LKISC_RSBTYPE = 00000002
LKISC_WAITING = 00000002
LKISSEARCH_BLOCKEDBY = 0000060D RG 02
LKISSEARCH_BLOCKING = 0000056C RG 02
LKISSND_BLKBY = ***** X 02
LKISSND_BLKING = ***** X 02
LKISSND_LOCKS = ***** X 02
LKISSND_STDREQ = ***** X 02
LKISV_SYSNAM = 0000001F
LKIS_BLOCKEDBY = 00000206
LKIS_BLOCKING = 00000207
LKIS_LASTLKB = 00000106
LKIS_LASTRSB = 00000209
LKIS_LCKCOUNT = 00000205
LKIS_LCKREFCNT = 00000103
LKIS_LOCKID = 00000104
LKIS_LOCKS = 00000208
LKIS_NAMESPACE = 00000200
LKIS_PARENT = 00000102
LKIS_PID = 00000100
LKIS_REMLKID = 00000105
LKIS_RESNAM = 00000201
LKIS_RSBREFCNT = 00000202
LKIS_STATE = 00000101
LKIS_SYSTEM = 00000204
LKIS_VALBLK = 00000203
LKID = 00000008
LKI_ALLOCATE = 000006C9 R 02
LOCAL_SPACE = 00000000
LOCK_INFO = 0000042B R 02
MAXCOUNT = 00000000 R 02
MAXSTRUC = 00000002
MAX_LKB_ITEM = 00000005
MAX_RSB_ITEM = 00000008
MOVEIT = 0000025A R 02
PCBSL_JIB = 00000080
PCBSL_PID = 00000060
PCBSQ_PRIV = 00000084
PCBSW_ASTCNT = 00000038
PCBSW_GRP = 000000BE
PR$ IPL = 00000012
PRVSV_SYSLCK = 0000001E
PRVSV_WORLD = 00000010
PSL$C_EXEC = 00000001
PSL$C_KERNEL = 00000000

```

SYS
V04-

SYSGETLKI
Symbol table

- GET LOCK MANAGER INFORMATION SYSTEM SE 16-SEP-1984 02:18:11 VAX/VMS Macro V04-00
5-SEP-1984 03:53:51 [SYS.SRC]SYSGETLKI.MAR;1

Page 37
(12)

```

PSL$$_PRVMOD      = 00000002
PSL$$_PRVMOD      = 00000016
RESERV            = 0000001C
RSB$$_RMOD        = 0000004E
RSB$$_RSNLEN      = 0000004F
RSB$$_CSID        = 00000038
RSB$$_CVTQFL      = 00000018
RSB$$_GRQFL       = 00000010
RSB$$_WTQFL       = 00000020
RSB$$_VALBLK      = 00000028
RSB$$_GROUP       = 0000004C
RSB$$_REFCNT      = 00000040
RSBTBC           = 00000026 R      02
SAVED_IPL         = FFFFFFFC
SCH$CREF          ***** X      02
SCH$GL_CURPCB     ***** X      02
SCH$POSTEF        ***** X      02
SPC_BLOCKEDBY     0000035D R      02
SPC_BLOCKING      00000392 R      02
SPC_LCKCOUNT     00000332 R      02
SPC_LOCKS         000003C7 R      02
SPC_NAMESPACE     00000321 R      02
SPC_PARENT        000002FD R      02
SPC_PID           000002D2 R      02
SPC_REMLKID       00000347 R      02
SPC_STATE         000002EB R      02
SPC_SYSTEM        0000030C R      02
SPECIAL          0000005C R      02
SPECIAL_LEN      = 0000000A
SS$_ACCVIO        = 0000000C
SS$_BADPARAM      = 00000014
SS$_BUFFEROVF     = 00000601
SS$_EXQUOTA       = 0000001C
SS$_INSFMEM       = 00000124
SS$_IVLOCKID      = 00002124
SS$_IVMODE        = 00000354
SS$_NOMORELOCK    = 00000A08
SS$_NORMAL        = 00000001
SS$_NOSYSLCK      = 000028F4
SS$_NOWORLD       = 00002884
SYSDCLAST         ***** GX    02
VALUE             = 00000000
VERIFYLOCKID      000004E4 R      02

```

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
.ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000030 (48.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
WSYSGETLKI	00000716 (1814.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
YEXEPAGED	00000008 (8.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
-----	-----	-----	-----
Initialization	34	00:00:00.07	00:00:00.38
Command processing	127	00:00:00.62	00:00:06.40
Pass 1	488	00:00:19.74	00:00:53.40
Symbol table sort	0	00:00:02.90	00:00:08.60
Pass 2	289	00:00:05.01	00:00:11.37
Symbol table output	19	00:00:00.16	00:00:00.37
Psect synopsis output	2	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	961	00:00:28.53	00:01:20.55

The working set limit was 2100 pages.
113868 bytes (223 pages) of virtual memory were used to buffer the intermediate code.
There were 100 pages of symbol table space allocated to hold 1809 non-local and 102 local symbols.
1661 source lines were read in Pass 1, producing 22 object records in Pass 2.
39 pages of virtual memory were used to define 38 macros.

! Macro library statistics !

Macro library name	Macros defined
-----	-----
_\$255\$DUA28:[SHRLIB]CLUSTER.MLB;1	1
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	18
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	13
TOTALS (all libraries)	32

1957 GETS were required to define 32 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:SYSGETLKI/OBJ=OBJ\$:SYSGETLKI MSRC\$:SYSGETLKI/UPDATE=(ENH\$:SYSGETLKI)+EXECMLS/LIB+SHRLIB\$:CLUSTER/LIB

0385

AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY